МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко

підпис

«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Ігровий застосунок в жанрі Survival Horror Stalker Shadow Of Pripyat на основі рушія Unreal Engine 5

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.22010913

Здобувач _____Н. О. Корнюков

підпис

«__» _____ 2024 р.

Керівник ст. викладач кафедри ІПЗ _____С. Ю. Боровльова

підпис

«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва

підпис

«__» _____ 2024 р.

Миколаїв – 2024

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

інженерії програмного

забезпечення, канд.

техн. наук, доцент,

____Є. О. Давиденко

«29» листопада 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

Корнюкову Нікіті Олеговичу

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

«Ігровий застосунок в жанрі Survival Horror Stalker Shadow Of Pripyat на основі рушія Unreal Engine 5»

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № 269

2. Строк представлення кваліфікаційної роботи «___» _____ 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є ігровий застосунок в жанрі Survival Horror Stalker Shadow Of Pripyat на основі рушія Unreal Engine 5».

4. Перелік питань, що підлягають розробці

Аналіз предметної області, порівняльний аналіз аналогів, визначення вимог та функціоналу системи, моделювання, проєктування програмного застосунку, кодування та тестування застосунку.

5. Перелік графічних матеріалів:

Презентація

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А.О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи

ст. викладач Боровльова Світлана Юріївна

(посада, прізвище, ім'я, по батькові)

(підпис)

Завдання прийнято до виконання

Корнюков Нікіта Олегович

(прізвище, ім'я, по батькові студента)

(підпис)

Дата видачі завдання «29» листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: «Ігровий застосунок в жанрі Survival Horror Stalker Shadow Of Pripyat на основі рушія Unreal Engine 5».

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	29.11.2023р.	30.11.2023р.	виконано
2.	Огляд літератури за темою роботи	15.02.2024р.	27.02.2024р.	виконано
3.	Складання календарного плану КРБ	28.02.2024р.	29.02.2024р.	виконано
4.	Аналіз предметної області	01.03.2024р.	15.03.2024р.	виконано
5.	Моделювання та конструювання ПЗ	15.03.2024р.	04.04.2024р.	виконано
6.	Кодування, тестування розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	05.04.2024р.	25.05.2024	виконано
7.	Розробка спеціальної частини з охорони праці	26.05.2024	30.05.2024	виконано
8.	Відгук керівника КРБ	01.06.2024	01.06.2024	виконано
9.	Оформлення КРБ та презентації	01.06.2024	02.06.2024	виконано
10.	Попередній захист	04.06.2024	04.06.2024	виконано
11.	Рецензування	11.06.2024	11.06.2024	виконано
12.	Завершення оформлення КРБ та презентації	12.06.2024	12.06.2024	виконано
13.	Захист кваліфікаційної роботи	26.06.2024	26.06.2024	

Розробив студент Корнюков Нікіта Олегович _____
(прізвище, ім'я, по батькові) (*підпис*)

«___»_____ 2024 р.

Керівник роботи ст. викладач Боровльова Світлана Юріївна _____
(посада, прізвище, ім'я, по батькові) (*підпис*)

«___»_____ 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Ігровий застосунок в жанрі Survival Horror Stalker Shadow Of Pripyat на основі рушія Unreal Engine 5»

Студент 409 гр.: Корнюков Нікіта Олегович

Керівник: ст. викладачка Боровльова Світлана Юріївна

Захищена: «__» 2024 р.

Кваліфікаційна робота бакалавра, присвячена розробці ігрового застосунку в жанрі Survival Horror Stalker Shadow Of Pripyat на основі ігрового рушія Unreal Engine 5, для популяризації жанру Survival Horror.

Об'єкт роботи: процеси, що пов’язані із організацією та створенням ігрового застосунку.

Предмет роботи: інструментальні засоби та інформаційні технології розробки ігрового застосунку.

Метою кваліфікаційної роботи є створення ігрового застосунку на основі Unreal Engine 5 для популяризації жанру Survival Horror сучасному ринку.

Для досягнення поставленої цілі необхідно виконати наступні завдання:

– провести аналіз існуючих ігор на основі unreal engine 5, з акцентом на ігри в жанрі survival horror;

– вивчити можливості unreal engine 5 для створення ігрового середовища, персонажів, зброї та інших ігрових елементів;

– розробити ігровий дизайн-документ, який буде описувати концепцію гри, ігрові механіки, сюжет, персонажів, ігровий світ та інші аспекти;

– створити 3d-моделі персонажів, зброї та інших ігрових елементів;

– запрограмувати ігрову логіку, систему взаємодії з ігровим світом та інші компоненти гри;

– виконати моделювання програмного забезпечення;

– створити ігровий рівень, який буде демонструвати основні ігрові механіки;

– провести тестування застосунку та зробити необхідні правки.

У вступі наводиться актуальність теми, а також визначаються мета та завдання для реалізації програмного застосунку

У першому розділі проведено аналіз існуючих аналогів, що дозволить визначити основні особливості ігор та сформувати загальне розуміння предметної області.

У другому розділі описано процес розробки проєктних рішень з метою відповіді на вимоги до програмного забезпечення. Цей процес включає розробку UML-діаграм та опис інтерфейсів.

У третьому розділі показано процес розробки, включаючи вибір мови програмування, вибір рушія.

У четвертому розділі описано розробку геймплею гри, та розробку ігрового рівня.

У висновках проводиться аналіз роботи та отриманих результатів.

КРБ викладена на 68 сторінок, вона містить 4 розділи, 49 ілюстрацій, 8 таблиць, 22 джерела в переліку посилань.

Ключові слова: Unreal Engine, horror, survival, C++, UML-діаграми

ABSTRACT

to the bachelor's qualification work

«Game application in the genre of Survival Horror Stalker Shadow Of Pripyat based on the Unreal Engine 5»

Student 409 gr.: Kornyukov Nikita Olegovich

Supervisor: senior lecturer Borovleva Svetlana Yuryevna

Summary of work:

A bachelor's thesis dedicated to the development of a Survival Horror game application Stalker Shadow Of Pripyat based on the Unreal Engine 5 game engine to popularize the Survival Horror genre.

Objective: to create a game application based on Unreal Engine 5 to popularize the Survival Horror genre in the modern market.

Object of work: processes related to the organization and creation of a game application.

Subject of work: tools and information technologies for developing a game application.

To achieve this goal, the following tasks need to be accomplished:

– to analyze existing games based on unreal engine 5, with an emphasis on survival horror games;

– to study the capabilities of unreal engine 5 for creating game environments, characters, weapons and other game elements;

– to develop a game design document that will describe the game concept, game mechanics, plot, characters, game world and other aspects;

– create 3d models of characters, weapons and other game elements;

– program the game logic, system of interaction with the game world and other game components;

– perform software modeling;

– create a game level that will demonstrate the basic game mechanics;

– test the game and make the necessary corrections.

The introduction defines the relevance of the topic, explains the purpose of the study and provides a brief overview of the task, subject of study and object of study.

The first section analyzes existing analogs to identify the main features of games and form a general understanding of the subject area.

The second section describes the process of developing design solutions to meet software requirements. This process include the development of UML diagrams and the description of interfaces.

The third section shows the development process, including choosing a programming language, selecting an engine, designing its gameplay, and developing a game level.

The fourth section describes the development of the game's gameplay and the development of the game level.

The bachelor's qualification work is laid out on 68 page, it contains 4 sections, 49 illustrations, 8 tables, 22 sources in the list of references.

Keywords: Unreal Engine, horror, survival, C++, UML–diagrams

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

API – application programming interface

UE5 – Unreal Engine 5

UI – user interface

UML – unified modeling language

ВСТУП

Актуальність обраної теми.

В сучасному світі ігри стають популярнішими з кожним днем. Тому ігри розвиваються буквально щодня. Unreal Engine 5 – це сучасний ігровий двигун, який використовується для створення високоякісних ігор з реалістичною графікою та фізикою. Він набуває все більшої популярності серед розробників ігор завдяки своїм широким можливостям та простоті використання. На українському ринку існує дефіцит якісних комп'ютерних ігор в стилі Survival Horror. Кваліфікаційна робота бакалавра робота передбачає розробку ігрового застосунку на основі Unreal Engine 5.

Об’єкт кваліфікаційної роботи – процеси, що пов’язані із організацією та створенням ігрового застосунку.

Предмет кваліфікаційної роботи – інструментальні засоби та інформаційні технології розробки ігрового застосунку.

Метою кваліфікаційної роботи є створення ігрового застосунку на основі Unreal Engine 5, для популяризації жанру Survival Horror сучасному ринку.

Для досягнення цієї мети необхідно вирішити наступні завдання:

– провести аналіз існуючих ігор на основі unreal engine 5, з акцентом на ігри в жанрі survival horror;

– вивчити можливості unreal engine 5 для створення ігрового середовища, персонажів, зброї та інших ігрових елементів;

– розробити ігровий дизайн-документ, який буде описувати концепцію гри, ігрові механіки, сюжет, персонажів, ігровий світ та інші аспекти;

– створити 3d-моделі персонажів, зброї та інших ігрових елементів;

– запрограмувати ігрову логіку, систему взаємодії з ігровим світом та інші компоненти гри;

– виконати моделювання програмного забезпечення;

– створити ігровий рівень, який буде демонструвати основні ігрові механіки;

– провести тестування застосунку та зробити необхідні правки.

Очікувані результати:

– розроблений ігровий застосунок на основі unreal engine 5;

– 3d-моделі персонажів, зброї та інших ігрових елементів;

– запрограмована ігрова логіка, система взаємодії з ігровим світом та інші компоненти гри;

– ігровий рівень, який буде демонструвати основні ігрові механіки;

– звіт про тестування застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Огляд предметної області

В сучасному світі ігри стали невід'ємною частиною життя людей різного віку, професій та соціального статусу. Їхня популярність стрімко зростає, що обумовлюється декількома факторами:

– доступність – завдяки розвитку технологій та поширенню комп'ютерів, смартфонів, планшетів та інших ігрових платформ, ігри стали доступними для більшої кількості людей;

– різноманіття – існує безліч різних жанрів ігор, що дозволяє кожному знайти гру, яка відповідає його інтересам;

– зміна ставлення – раніше ігри сприймалися як марна трата часу. сьогодні ж ігри визнані як цінний інструмент для освіти, навчання та розвитку;

– поява нових технологій – нові технології, такі як віртуальна реальність та доповнена реальність, роблять ігри більш реалістичними та захоплюючими.

Жанр «Survival Horror» є одним з найцікавіших і популярних жанрів в індустрії відеоігор. Він відрізняється своєю унікальною атмосферою та геймплеєм, спрямованим на створення напруженої та страшної атмосфери, де гравець змушений боротися за виживання у вражаючих та часто ворожих умовах. Основні концепції та поняття, пов'язані з жанром «Survival Horror», включають:

– виживання – одна з основних ідеї жанру полягає в тому, щоб гравцеві виживати в небезпечному середовищі, де ресурси, такі як здоров'я, боєприпаси та інші необхідні предмети, обмежені;

– жах – жах є ключовим елементом жанру. це може бути досягнуто за допомогою страшних монстрів, зловісних місць, напруженого звукового оформлення та інших елементів, які створюють напружену та неспокійну атмосферу;

– обмежені ресурси – гравцю часто доводиться обмежувати використання своїх ресурсів, таких як боєприпаси, медикаменти та інше, щоб вижити. це створює додатковий напружений елемент гри;

– пазлові елементи – багато ігор у жанрі «survival horror» містять пазлові елементи, де гравцеві потрібно розв'язувати загадки або знаходити шляхи виходу з складних ситуацій;

– атмосфера – створення відповідної атмосфери є ключовим для успіху гри в цьому жанрі. це може включати звукове оформлення, освітлення, дизайн рівнів та інші елементи, які допомагають створити напружену та неспокійну атмосферу.

Кожен рік розробляється велика кількість різноманітних ігор, деякі ігрові компанії створюють власні двигуни для своїх ігор, а деякі компанії використовують вже готові ігрові двигуни, наприклад: Unity, UE5, Godot.

Unreal Engine 5 надає безліч можливостей для розробки ігор, включаючи потужну графіку, динамічне освітлення та вражаючі візуальні ефекти, які можуть значно полегшити процес розробки та сприяти швидкому створенню вражаючих ігор.

Ось деякі аспекти які знадобляться для розробки ігор на Unreal Engine 5:

– моделювання та анімація – для створення вражаючої графіки потрібні навички в області 3d моделювання, текстурування, анімації та освітлення;

– програмування – знання програмування важливе для створення складних геймплейних механік, інтеграції штучного інтелекту та створення користувацьких скриптів. розробники повинні володіти мовами програмування, такими як c++, blueprint;

– дизайн геймплею – розробники повинні мати розуміння процесу створення геймплею, включаючи розробку рівнів, балансу гри, механіки гравця та інші аспекти геймдизайну;

– звуковий дизайн – звук грає важливу роль у створенні імерсійного ігрового досвіду. розробники повинні мати навички в області звукового дизайну та музичного складання, а також розуміння просторового звуку та атмосферного звукового оформлення;

– тестування та оптимізація – важливо проводити тестування гри на різних платформах та пристроях, а також оптимізувати її для досягнення плавної роботи та високої продуктивності.

Приклад комп'ютерних ігор у жанрі Survival Horror: S.T.A.L.K.E.R., STALKER on UE, S.T.A.L.K.E.R.: True Stalker.

1.2 Аналіз аналогів ігор жанру «Survival Horror»

Аналіз аналогів ігор жанру «Survival Horror» є важливим кроком у процесі розробки, оскільки дозволяє виявити як сильні, так і слабкі сторони існуючих ігор, а також визначити нішу для нового проєкту. Це допомагає уникнути повторення помилок, виявлення недоліків та підвищення якості гри, що розробляється.

Таблиця 1.1 – Опис S.T.A.L.K.E.R.

Назва	S.T.A.L.K.E.R. [1]
Виробник	GSC Game World
Архітектура	PC (Microsoft Windows)
Мова реалізації	C++
Функції та характеристики	– відкритий світ: гра пропонує реалістичний і великий открытий світ, що дозволяє гравцям вільно досліджувати постапокаліптичне оточення зони; – аномалії та мутанти: унікальні аномалії та мутанти в зоні створюють непередбачуване геймплейове середовище;

Кінець таблиці 1.1

Переваги	– атмосфера: унікальна атмосфера гри, яка створює напружене та містичне враження; – відкритий світ: широкі можливості для вільного дослідження та взаємодії з навколишнім середовищем; – реалістичність: реалістична поведінка аномалій, мутантів та взаємодія з оточенням.
Недоліки	–технічні проблеми: у деяких версіях гри можуть виникати технічні проблеми, такі як баги та збої; –графіка: у ранніх версіях гри графіка може виглядати застарілою порівняно з сучасними стандартами.
Приклад зображень	 Рисунок 1.1 – Зображення геймплею
Джерело інформації	https://stalker.fandom.com/uk/wiki/S.T.A.L.K.E.R.:_Поклик_Прип’яті

Таким чином було проаналізовано S.T.A.L.K.E.R. було виділено недоліки та переваги, і далі буде розглядатися STALKER on UE.

Таблиця 1.2 – Опис STALKER on UE.

Назва	STALKER on UE [2]
Виробник	GSC Game World, ентузіасти

Кінець таблиці 1.2

Архітектура	PC (Microsoft Windows)
Мова реалізації	C++
Функції та характеристики	– відкритий світ: гра пропонує реалістичний і великий открытий світ, що дозволяє гравцям вільно досліджувати постапокаліптичне оточення зони; – аномалії та мутанти: унікальні аномалії та мутанти в зоні створюють непередбачуване та небезпечне геймплейове середовище; – торгівля: гравці можуть здійснювати торгівлю, знаходити та модифікувати зброю, а також взаємодіяти з іншими сталкерами для досягнення своїх цілей.
Переваги	– новий сюжет, дія якого відбувається після подій Поклик Прип'яті, але зав'язана на природних історіях та нових персонажах; – відкритий світ: широкі можливості для вільного дослідження та взаємодії з навколишнім середовищем; – реалістичність: реалістична поведінка аномалій, мутантів та взаємодія з оточенням; – графіка: нове освітлення.
Недоліки	–збереження: можливість зберігати гру не були реалізовані; –графіка: текстури та моделі залишились як в оригінальній трилогії; –сюжет: сюжет був взятий з оригінальної трилогії без змін.

Кінець таблиці 1.2

Приклад зображень	 Рисунок 1.2 – Зображення геймплею
Джерело інформації	https://s2ue.org/en

Таким чином було проаналізовано STALKER on UE було виділено недоліки та переваги, і далі буде розглядатися S.T.A.L.K.E.R. True Stalker.

Таблиця 1.3 - Опис S.T.A.L.K.E.R.: True Stalker.

Назва	S.T.A.L.K.E.R.: True Stalker [3]
Виробник	GSC Game World, ентузіасти
Архітектура	PC (Microsoft Windows)
Мова реалізації	C++
Функції та характеристики	– відкритий світ: гра пропонує реалістичний і великий открытий світ, – аномалії та мутанти: унікальні аномалії та мутанти в зоні створюють непередбачуване геймплейове середовище; – торгівля: гравці можуть здійснювати торгівлю, знаходити та модифікувати зброю, а також взаємодіяти з іншими сталкерами для досягнення своїх цілей;

Кінець таблиці 1.3

Переваги	– новий сюжет: сюжет дія якого відбувається після подій поклик прип'яті, але зав'язана на природних історіях та нових персонажах; – нові особливості: завдання гри супроводжуються кат-сценами, авторським озвученням персонажів, а в деяких випадках і варіативністю. завдання в грі не мають ухилу; – нові локації: 13 ігрових локацій, серед яких вже найкращі, але покращені рівні трилогії s.t.a.l.k.e.r., а також деякі авторські локації.
Недоліки	–технічні проблеми: гра може вилітати, а також є інші баги з різними аспектами геймплея; –графіка: текстури та моделі залишились як в оригінальній трилогії.
Приклад зображень	 Рисунок 1.3 – Зображення геймплею
Джерело інформації	https://www.unian.ua/games/stalker-modi-dlya-poklik-prip-yati-viyshov-velikiy-mod-z-novim-syuzhetom-12491772.html

1.3 Специфікація вимог до ПЗ

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначення програмного забезпечення - створення захоплюючого та імерсійного ігрового досвіду в жанрі «Survival Horror». Гра призначена для розваги гравців, які цікавляться напруженими та захоплюючими ігровими пригодами, де вони змушені боротися за виживання в жорстоких та небезпечних умовах.

Погодження, що ухвалені в програмній документації

Було погоджено, що для створення гри буде використано ігровий рушій Unreal Engine 5.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 13.06.2024р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Застосунок призначений для розваги користувача, який цікавиться напруженими імерсійними ігровими дослідами, які дозволяють йому відчути атмосферу страху та виживання.

Характеристика користувачів

Основні характеристики користувачів: наявність ПК.

Загальні обмеження

Обмежень немає.

ФУНКЦІЇ СИСТЕМИ

Функція отримання досягнень

Опис функції

Функція досягнень створює додатковий стимул для гравців до виконання певних завдань або досягнення певних результатів в грі.

Вхідна і вихідна інформація

Вхідна інформація – назви локації, виконанні завдання, вихідна локація – отримання досягнення у головному меню.

Функціональні вимоги

Файл з досягненнями, які виконанні повністю або не до кінця.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

Основним джерелом вхідної інформації є користувач.

Нормативно-довідкова інформація (класифікатори, довідники тощо)

Вимоги відсутні.

Вимоги до способів організації, збереження та ведення інформації

Усі данні зберігаються в файл збереження JSON.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯді

– ОС: Windows 7 і вище

– Оперативна пам’ять: 8 GB

– Відеокарта: відео пам’ять 2 GB і більше

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Архітектура програмної системи складається з клієнтської частини.

Системне програмне забезпечення

Гра розроблена на платформі Unreal Engine 5 з використанням мови програмування C++.

Мережне програмне забезпечення

Для створення програмного забезпечення було використано ОС Windows 10.

Мова і технологія розробки ПЗ

Гра була розроблена на платформі Unreal Engine 5 з використанням мови програмування C++.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс має задовольняти усі вимоги дизайну, він є доволі зручним у використанні.

Апаратний інтерфейс

Апаратним інтерфейсом є ПК користувача, з операційною системою Windows 10.

Програмний інтерфейс

У ході розробки було використано дві категорії Unreal Engine API: UE5 Audio API, UE5 UI API.

Комунікаційний протокол

Відсутній.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Гра є доступною для будь-якого користувача, за умови наявності апаратного інтерфейсу.

Супроводжуваність

Гра не потребує супроводжуваності.

Переносимість

Програмне забезпечення може працювати на ОС Windows (7 версія та вище).

Продуктивність

Продуктивність роботи ПЗ залежить від характеристик ПК.

Надійність

Відсутня.

Безпека

Відсутня.

ІНШІ ВИМОГИ

Усі вимоги сформовано та описано вище, доповнення не вимагається.

Висновки до розділу 1

Під час формування першого розділу було закріплено знання та навички стосовно розробки ігор для ПК на базі ігрового рушія Unreal Engine 5. Було проведено аналіз предметної області застосунку, та визначено актуальність роботи. Акцент був зроблений на розгляді загального стану ігрової індустрії в світі та факторів, що призводять до її стрімкого зростання популярності. Висвітлено ключові причини, які роблять ігри на сьогоднішній день невід'ємною частиною життя для людей різного віку та соціального статусу.

Додатково було проведено детальний огляд жанру «Survival Horror», в якому були визначені основні концепції та поняття, що визначають цей жанр, а також наведено приклади ігор, які є представниками цього жанру. Зокрема, висвітлено важливі аспекти такі як виживання, жах, обмеженість ресурсів, пазлові елементи та створення відповідної атмосфери.

Розглянуто також ключові аспекти, які знадобляться для розробки ігор на платформі Unreal Engine 5, такі як моделювання та анімація, програмування, дизайн геймплею, звуковий дизайн та тестування та оптимізація.

Після цього було проведено аналіз існуючих застосунків-аналогів, було проаналізовано мови реалізації, функції, характеристики та виділено основні переваги та недоліки.

Також в розділі визначено специфікацію вимог до ПЗ, що розробляється.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

Моделювання програмного застосунку є ключовою складовою при проєктуванні та розробці програмного забезпечення. Цей розділ визначає основні концептуальні та архітектурні аспекти створення програмного застосунку і відповідає на питання про те, як програма буде працювати, які будуть її основні функції та які вимоги вона повинна задовольняти.

2.1 Створення сценаріїв використання

Короткий usecase

Користувач запускає гру та завантажує збереження. Система завантажує користувача в ігровий світ та розташовує його в базовому таборі сталкерів. Користувач отримує повідомлення від торговця щодо нового завдання - дослідження аномалій. Користувач вирушає у відповідь на завдання. В дорозі він зустрічає інших сталкерів, можливі ворожі фракції чи небезпечних монстрів. Користувач успішно доходить до місця завдання, активує пристрій виявлення аномалій та отримує цінні артефакти. Після завершення завдання Користувач повертається в базовий табір, де здає завдання торговцю та обмінює артефакти на ресурси. Система пропонує користувачу вибрати наступне завдання або завершити гру. Користувач обирає «завершити гру», зберігає свій прогрес та виходить з гри.

Поверхневий usecase

Головний сценарій (успішний):

Користувач входить в гру і завантажує збереження. Користувач підходить до торговця, гра перевіряє кількість грошей та репутацію. Користувачу виводить товари в залежності від репутації та грошей. Система видає унікальні ідентифікатори для вибраних предметів, які додаються до інвентаря користувача. При цьому користувач може використовувати придбані предмети у грі.

Альтернативні сценарії:

1) користувач не має достатньо грошей для придбання обраних товарів;

2) торговець не має в наявності обраних товарів через випадкові події у грі, наприклад мутанти напали на торгівельний путь;

3) користувач використовує чит для отримання необмеженої кількості грошей. система виявляє цей шахрайський вплив і блокує користувача;

4) виникає технічний збій у системі гри, і торговець не може завантажити інвентар;

5) користувач вирішує використати спритність та підступність, намагаючись вкрасти артефакти без оплати. торговець помічає спробу крадіжки та викликає охорону.

Таблиця 2.1 – Повний usecase

Scope	система купівлі, продажу в грі
Level	User-goal
Primary Actor	Користувач, торговець
Stakeholders and interests	1) користувач: Має інтерес придбати артефакти та покращення для поліпшення геймплею; 2) торговець: Має інтерес продавати товари та отримувати гроші від користувача.
Preconditions	1) користувач: зацікавлений в придбані товарів; 2) користувач: має достатню кількість грошей для придбання артефактів; Success guarantee: 1) користувач отримує придбані артефакти та покращення у свій інвентар.
Main Success Scenario: 1) користувач підходить до торговця; 2) Користувач натискає на нього для взаємодії;	

Кінець таблиці 2.1

3) система відображає інтерфейс торговця з варіантами покупки; 4) торговець вивчає інвентар користувача та його кількість грошей; 5) користувач переглядає асортимент артефактів та покращень; 6) користувач вибирає конкретний артефакт або покращення для придбання; 7) користувач натискає «придбати» ; 8) система генерує унікальні ідентифікатори для купленого товару та додає їх до інвентаря користувача; 9) користувач отримує підтвердження покупки та зміну стану грошового балансу; 10) користувач виходить з вікна діалогу; 11) торговець висловлює подяку.	
Extensions	1) користувач має недостатню кількість грошей; а) торговець відмовляє в продажі та надає користувачу інші варіанти торгівлі; 2) торговець не має в наявності обраних артефактів; а) користувач може обрати інші доступні предмети або зачекати, поки товар знову з'явиться в наявності; 3) користувач продає свої товари а) користувач переміщає свої товари до вікна продажу; б) користувач натискає на кнопку продажу, і отримує гроші.
Special Requirements	1) забезпечити захист системи від читів для уникнення шахрайства.
Technology and Data Variations List	1) використання унікальних ідентифікаторів для предметів в інвентарі користувача. 2) кількість товару, що купується або продається, вводиться користувачем.

Кінець таблиці 2.1

Frequency of Occurrence	Середня частота виконання - кілька разів під час геймплею, залежно від стратегії користувача.
Miscellaneous	1) як реагувати на спроби шахрайства та читів? 2) чи повинен торговець реагувати на спробу користувача залізти за стойку торгівця?

2.2 Побудова діаграм сценаріїв використання

Діаграми сценаріїв використання є важливим інструментом при моделюванні програмного застосунку, оскільки вони дозволяють уявити та проаналізувати різні сценарії використання програми з точки зору кінцевого користувача. Основна мета діаграм сценаріїв використання полягає в тому, щоб зрозуміти, як користувачі будуть взаємодіяти з програмним забезпеченням та як воно буде використовуватися в реальних умовах.

Діаграми сценаріїв використання допомагають розробникам проєктувати ефективний та зручний інтерфейс користувача. Вони дозволяють ідентифікувати та враховувати потреби користувачів при створенні інтерфейсу.

Також діаграми сценаріїв використання можуть використовуватися для планування та виконання тестування програмного забезпечення. Вони дозволяють визначити набори тестових випадків, які покривають різні сценарії використання програми.

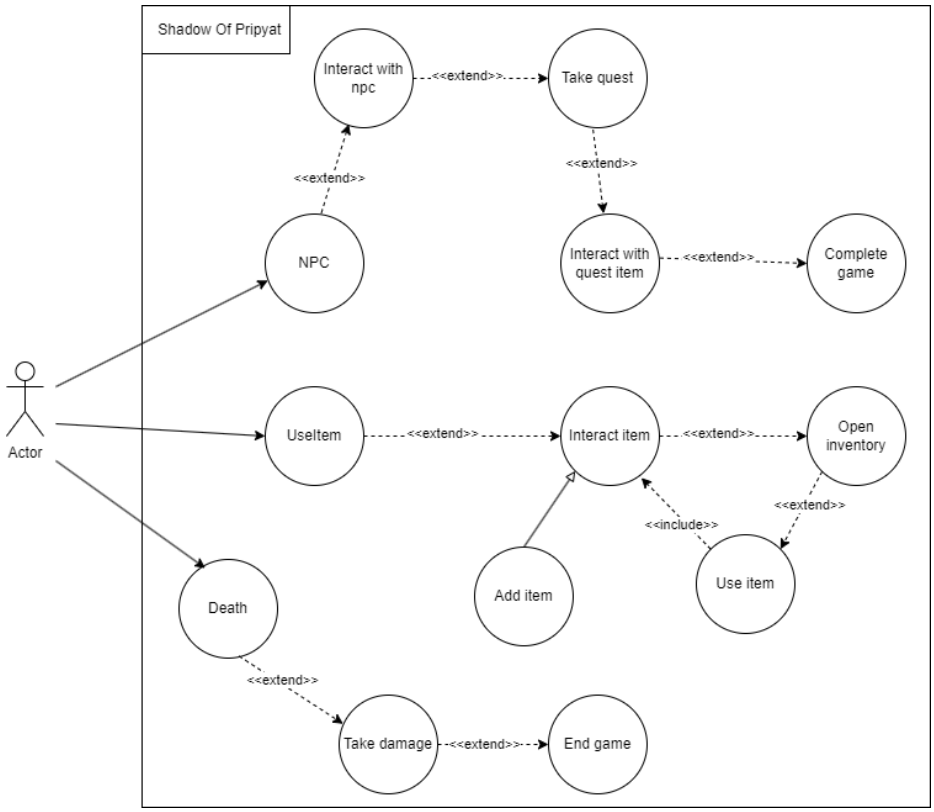

Рисунок 2.1 – Діаграма сценаріїв використання

На даній діаграмі зображено можливі сценарії гри, що охоплюють взаємодію гравця з іншими предметами, взаємодію з іншими персонажами, а також сценарії, пов'язані з подіями, які можуть відбутися з персонажем гравця.

2.3 Побудова діаграм станів

Діаграми станів є графічними моделями, які використовуються для опису поведінки системи відносно її різних станів та переходів між ними. Основне їх призначення - це візуалізація різних можливих станів системи та умов, за яких вона може змінювати свій стан.

Діаграми станів дозволяють визначити всі можливі стани, в яких може перебувати система, та умови, за яких вона переходить з одного стану в інший.

А також вони відображають умови, за яких система переходить з одного стану в інший, що допомагає розуміти, як система реагує на вхідні події або дії користувачів.

Для побудови діаграм станів спочатку треба визначити всі можливі стани системи. Потім ідентифікувати події або дії, що впливають на систему, та визначити переходи між станами для кожної події. Після цього потрібно визначити дії, пов'язані з кожним переходом, та треба побудувати саму діаграму, використовуючи символи та стрілки для позначення станів, переходів та дій.

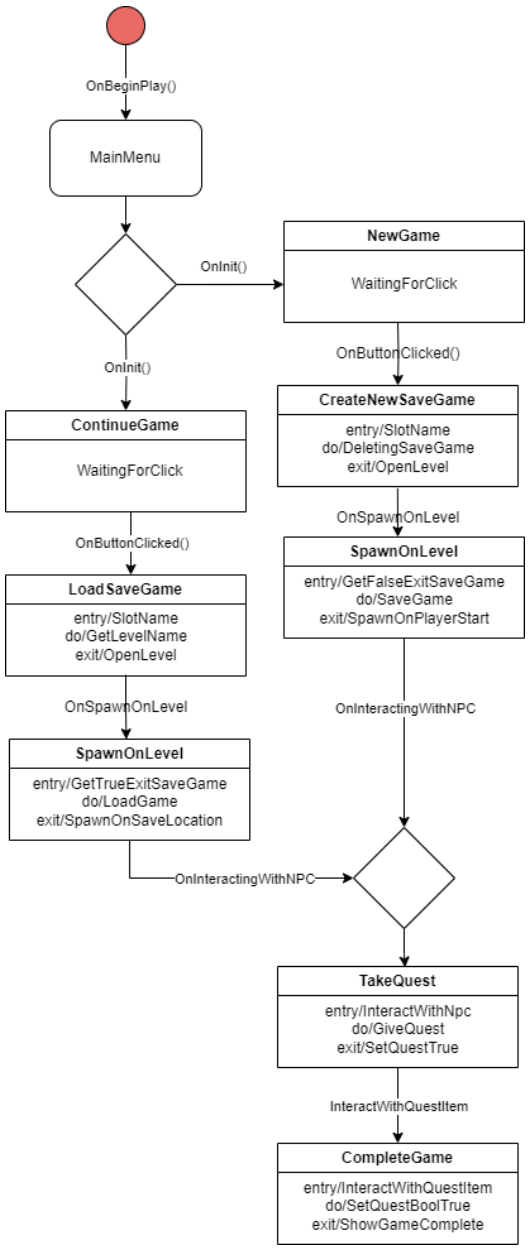

Рисунок 2.2 – Діаграма стану системи в цілому

Таким чином була побудована діаграма стану системи в цілому.

На діаграмі станів системи в цілому, відображено стани переходів через різні етапи гри. До цих етапів входить головне меню, початок нової гри або продовження вже існуючої. Після вибору нової гри або продовження гри, користувача завантажує на рівень, після чого користувач може пройти гру. Якщо гра пройдена то висвітиться результат.

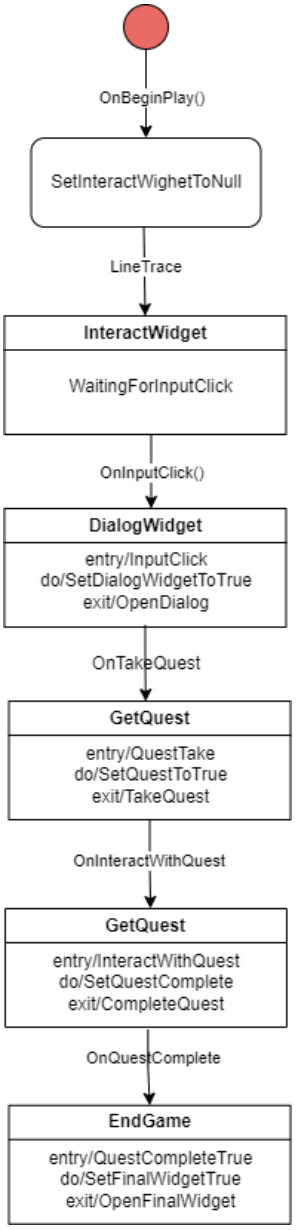

Рисунок 2.3 – Діаграма станів завдань

Таким чином була побудована діаграма станів завдань.

На другій діаграмі станів відображено елемент проходження завдання. Коли гравець проходить завдання, то гра завершується і висвічується фінальне вікно з результатом.

2.4 Побудова діаграм діяльностей

Діаграма діяльностей - це графічне зображення послідовності дій або процесу в системі або програмі. Вона використовується для моделювання та візуалізації послідовності подій або операцій, які відбуваються під час виконання конкретної діяльності. Діаграма діяльностей дозволяє зрозуміти, які кроки потрібно виконати та у якому порядку, а також які умови або варіанти можуть виникнути під час виконання цих дій. Цей тип діаграми часто використовується для аналізу та проєктування бізнес-процесів, програмного забезпечення та інших систем.

Для побудови діаграми діяльностей необхідно мати чітке розуміння процесу або діяльності, що моделюється. Далі використовуються стандартні символи і нотація для візуалізації послідовності дій та умов. Також важливо врахувати взаємодію між учасниками процесу та умови, які впливають на хід виконання.

Діаграма діяльностей складається з елементів, що відображають дії або діяльності, стрілок, що показують порядок виконання кроків, та різних видів вузлів, які вказують на умови, рішення та інші аспекти процесу. Кожен елемент відображає конкретну дію або стан, а стрілки показують послідовність цих дій.

На першій діаграмі зображено діалог з іншим персонажем. На рисунку 2.4 відображається вибір гравця, якщо гравець вибирає простий діалог то він просто отримує інформацію, а якщо обирає діалог з завданням, то після кінця діалогу йому видається завдання.

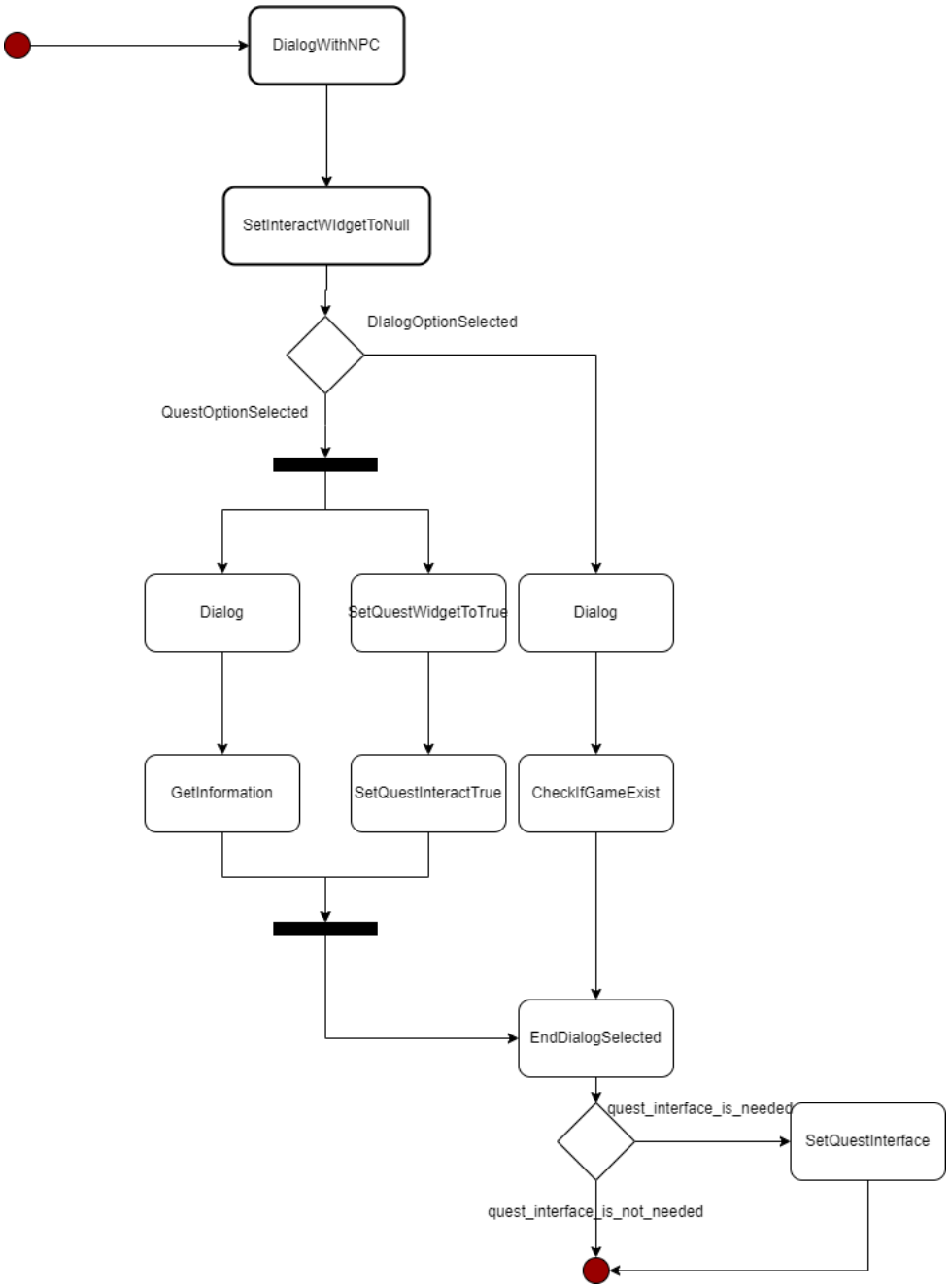

Рисунок 2.4 – Діаграма діяльності початку гри

Друга діаграма відображає початок гри. Гравець вибирає почати нову гру чи продовжити вже існуючу гру, після вибору нової гри запускається рівень, після цього він зберігається.

Якщо гравець вибрав продовжити гру, то гравець може вибрати слот який потрібно завантажити, після цього йдеться перевірка наявності збережень та у разі їх наявності запускається потрібний рівень.

Якщо збережень немає, то починається нова гра.

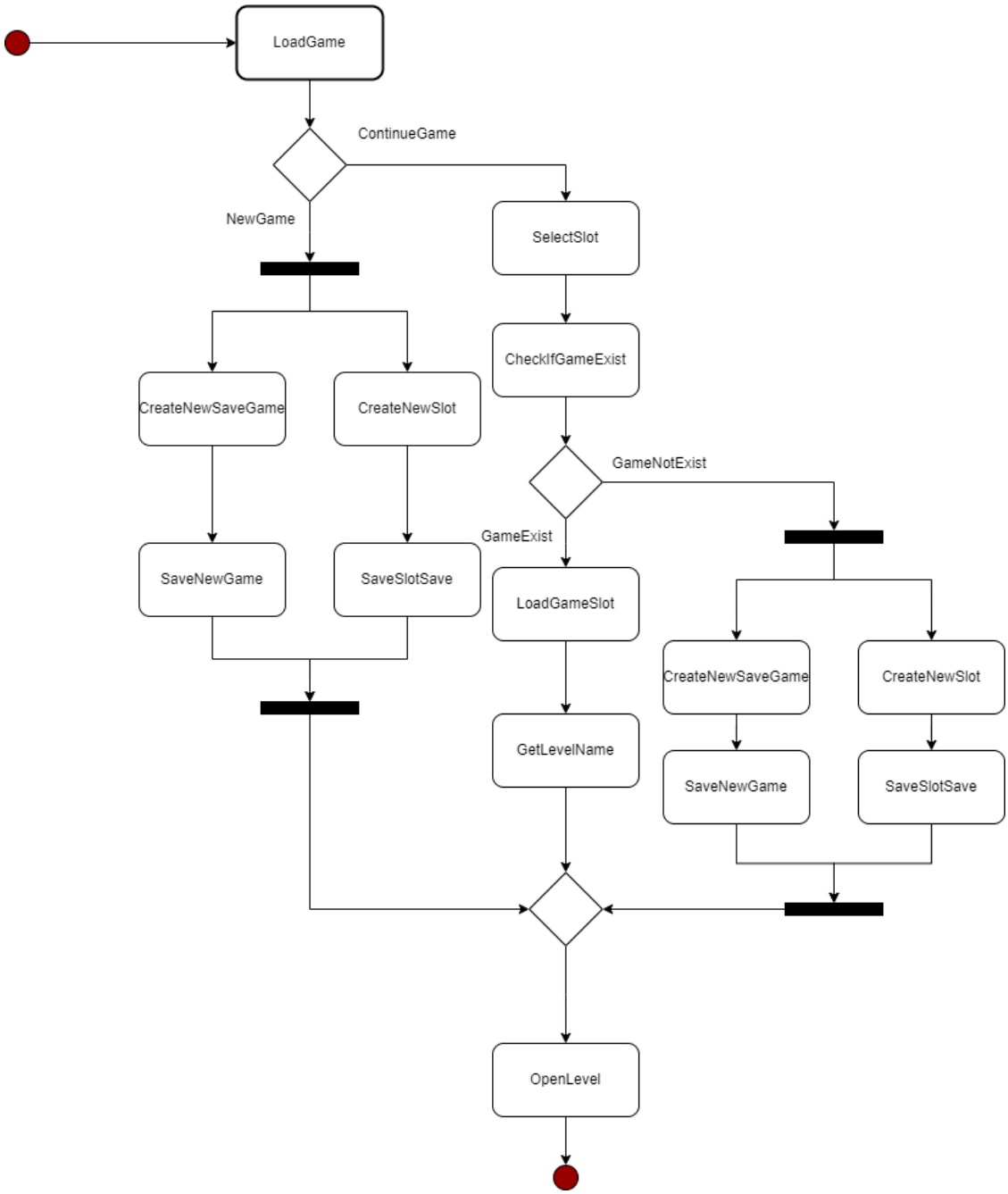

Рисунок 2.5 – Діаграма діяльності продовження гри

Таким чином була представлена діаграма діяльності яка показує систему продовження гри.

2.5 Побудова діаграми взаємодії

Діаграми взаємодії відображають взаємодію між різними об'єктами чи системами у вигляді послідовностей повідомлень чи дій. Вони дозволяють моделювати та аналізувати різноманітні сценарії взаємодії у системі. Ці діаграми зазвичай використовуються для опису взаємодії між користувачем та системою, а також між різними компонентами системи.

Існують кілька типів діаграм взаємодії, серед яких найпоширенішими є діаграми послідовності та діаграми співпраці. Діаграми послідовності показують послідовність повідомлень, що передаються між об'єктами, визначаючи порядок виконання операцій. Діаграми співпраці відображають взаємодію між об'єктами у термінах їх взаємних взаємодій та залежностей.

Діаграми взаємодії допомагають розуміти, як система повинна функціонувати та які комунікаційні шляхи між її складовими частинами. Вони є важливим інструментом під час аналізу та проєктування програмного забезпечення, оскільки дозволяють виявити можливі проблеми у взаємодії між компонентами системи та вдосконалити їх архітектуру.

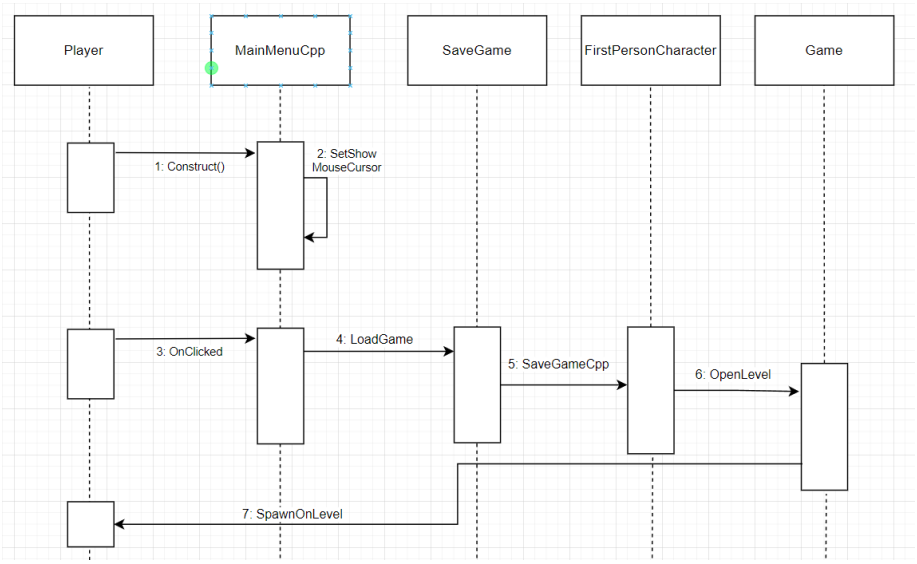

Рисунок 2.6 – Діаграма взаємодії

На діаграмі взаємодії показано взаємодію користувача з головним меню.

2.6 Створення mockup інтерфейсу застосунку

Mockup інтерфейсу застосунку - це прототип або макет інтерфейсу програмного застосунку, який створюється для візуалізації та тестування дизайну перед його фактичною реалізацією. Це спрощена версія інтерфейсу, яка дозволяє команді розробників та зацікавленим сторонам отримати уявлення про те, як виглядатиме програма та яким чином буде взаємодіяти з користувачем.

Mockup інтерфейсу може включати в себе елементи такі як кнопки, поля введення, меню, вікна та інші компоненти, які є необхідними для відтворення основного функціоналу програми. Вони зазвичай не мають функціональності, а лише демонструють вигляд та розташування елементів інтерфейсу.

Створення mockup дозволяє команді розробників та дизайнерам ефективно спілкуватися між собою та з клієнтом, зосереджуючись на вигляді та функціональності програми перед тим, як розпочати її реалізацію. Також це допомагає виявити потенційні проблеми з дизайном та взаємодією з користувачем на ранніх етапах розробки, що забезпечує швидше та ефективніше створення кінцевого продукту.

Перший мокап це мокап головного меню гри (рис. 2.7).

Рисунок 2.7 – Мокап головного меню гри

Головне меню гри. В ньому гравець зможе почати нову гру, продовжити вже існуючу гру. Також є кнопка налаштування, натиснувши яку гравець зможе перейти до налаштувань гри. А також є кнопка для виходу з гри на робочий стіл.

Другий мокап це мокап налаштувань гри (рис. 2.8)

Рисунок 2.8 – Мокап налаштувань в головному меню

В налаштуваннях гри, користувач зможе змінити повноекранний режим, роздільну здатність, заготовки якості а також динамічну роздільну здатність. А також може включити або виключити вертикальну синхронізацію і після цього користувач може застосувати зміни та вийти назад у головне меню.

Третім мокап був мокап інвентаря головного персонажа (рис. 2.9).

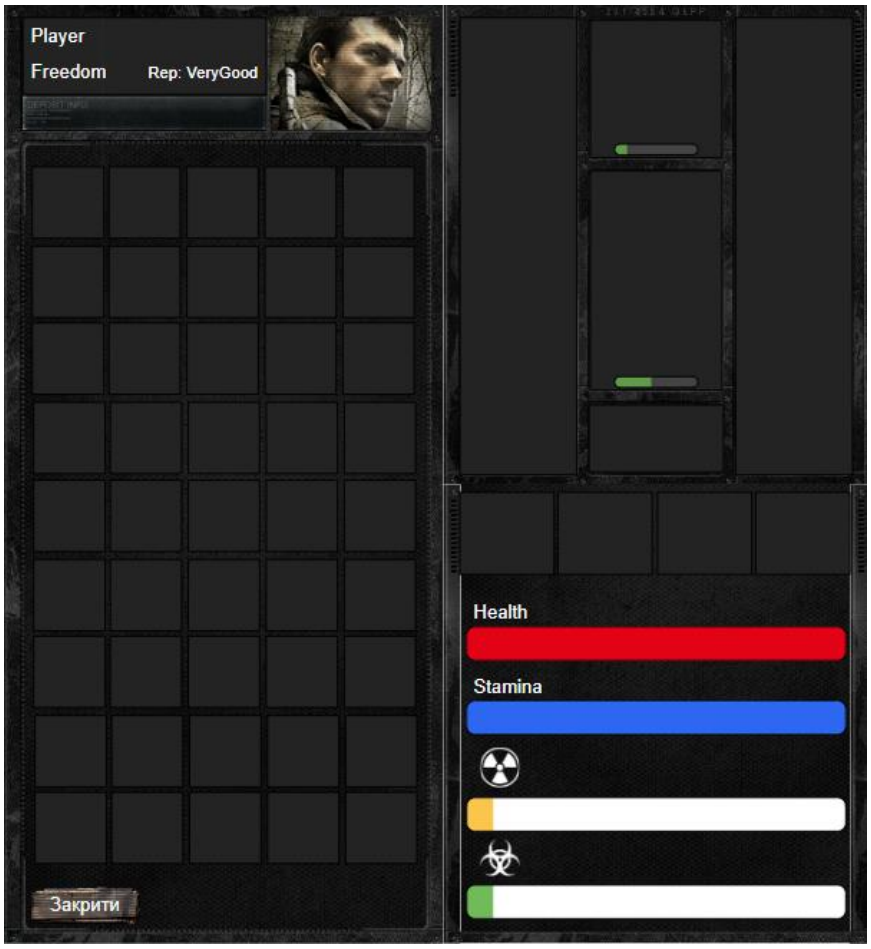

Рисунок 2.9 – Мокап інвентаря

В інвентарі можна буде використовувати різні предмети, змінювати одяг та зброю, слідкувати за станом здоров'я, витривалості а також за рівнем радіації та хімічний зараженням.

Висновки до розділу 2

У другому розділі було розглянуто процес створення різних типів UML-діаграм, зокрема діаграм станів, використання, взаємодії та діяльностей. Було створено власні діаграми, а також описано сценарії використання у різних формах, включаючи короткий, поверхневий та повний. Також було створено мокапи для гри що проєктується. На основі проведеного аналізу можна розпочинати розробку ігрового застосунку в жанрі Survival Horror.

3 ОГЛЯД СТЕКУ ЗАДІЯНИХ ТЕХНОЛОГІЙ ТА ВИБІР ІГРОВОГО РУШІЯ

3.1 Розгляд ігрових рушіїв

Перед початком створення програмного забезпечення, важливо ретельно розглянути доступні ігрові рушії. Вибір ігрового рушія є одним із найважливіших рішень при розробці ігрового застосунку. Ігровий рушій визначає основні можливості та обмеження проєкту, впливає на якість графіки, продуктивність, а також зручність роботи для розробників. Існує багато різних ігрових рушіїв, кожен з яких має свої переваги та недоліки.

Unreal Engine

Unreal Engine – це потужний ігровий рушій, розроблений компанією Epic Games[9]. Він відомий своєю високою продуктивністю, відмінною графікою та широкими можливостями для розробки ігор різних жанрів. Unreal Engine використовується як великими студіями, так і незалежними розробниками.

Рисунок 3.1 – Логотип «Unreal Engine 5»

Unreal Engine відомий своєю здатністю відтворювати графіку найвищої якості. Завдяки своїм потужним інструментам рендерингу, рушій дозволяє створювати реалістичні візуальні ефекти, що робить його ідеальним вибором для створення ААА–ігор.

Unreal Engine має велику та активну спільноту розробників, що забезпечує доступ до численних ресурсів, таких як навчальні матеріали, приклади коду, форуми та підтримка. Це значно полегшує процес навчання та вирішення проблем, що виникають під час розробки.

Рисунок 3.2 – Інтерфейс ігрового рушія «Unreal Engine 5»

Unity

Unity – це один з найпопулярніших ігрових рушіїв, розроблений компанією Unity Technologies[10]. Відомий своєю гнучкістю та простотою використання, Unity є ідеальним вибором для розробників різних рівнів – від початківців до професіоналів. Цей рушій широко використовується для створення 2D та 3D ігор, а також інтерактивних додатків.

Рисунок 3.3 – Логотип «Unity»

Unity пропонує багатий набір інструментів для розробки ігор, включаючи редактор сцени, анімаційний редактор, інструменти для створення графіки та звуку, а також потужні засоби для налаштування фізики та світла. Це дозволяє створювати складні та багатошарові ігрові проєкти.

Unity підходить як для невеликих інді-проєктів, так і для великих комерційних проєктів. Його можливості дозволяють створювати як прості 2D ігри, так і складні 3D світи з високою якістю графіки та детальною фізикою.

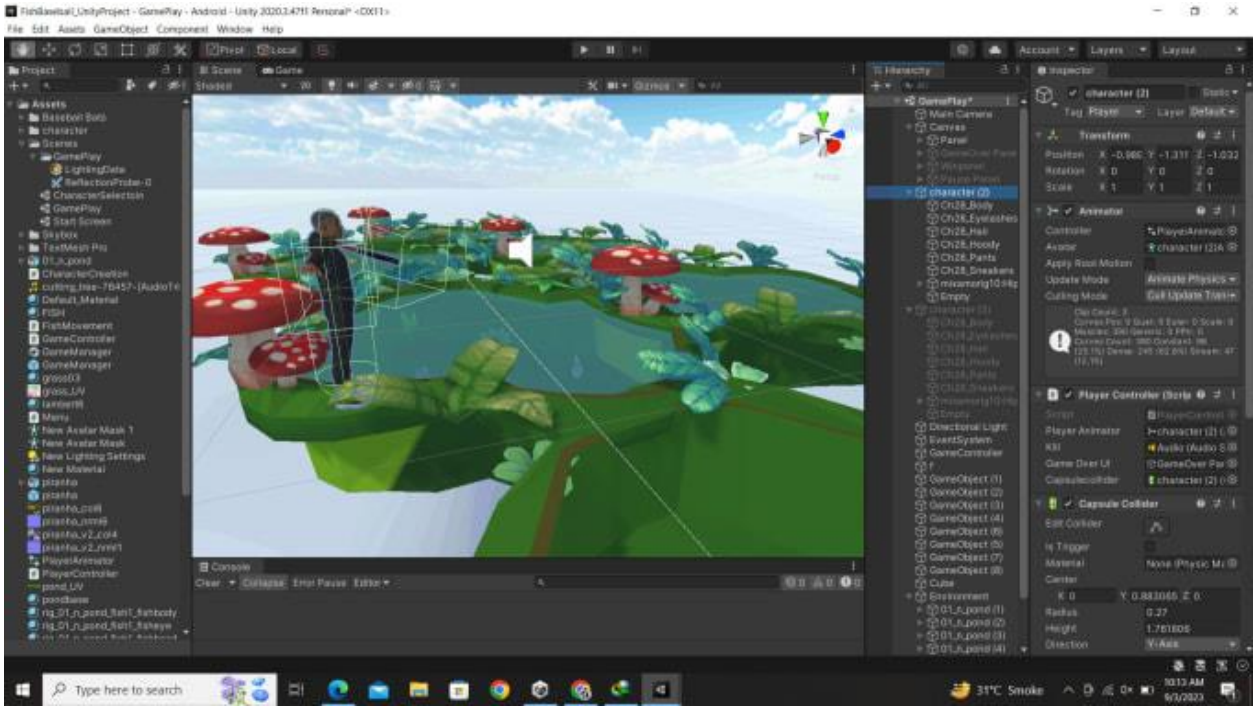

Рисунок 3.4 – Інтерфейс ігрового рушія «Unity»

CryEngine

CryEngine – це високопродуктивний ігровий рушій, розроблений компанією Crytek. Відомий своєю здатністю створювати графіку найвищої якості, CryEngine використовується для розробки різних ігор, включаючи шутери від першої особи, рольові ігри та відкриті світи [11].

Рисунок 3.5 – Логотип «CryEngine»

CryEngine забезпечує фото реалістичну графіку, використовуючи передові технології рендерингу. Це дозволяє створювати детальні світи з високою роздільною здатністю текстур, реалістичним освітленням і тінями, а також ефектами пост обробки.

CryEngine має потужний фізичний рушій, який дозволяє створювати реалістичну взаємодію об’єктів у грі, включаючи руйнування, рідини, тканини та інші фізичні ефекти. Це підвищує реалістичність ігрового процесу.

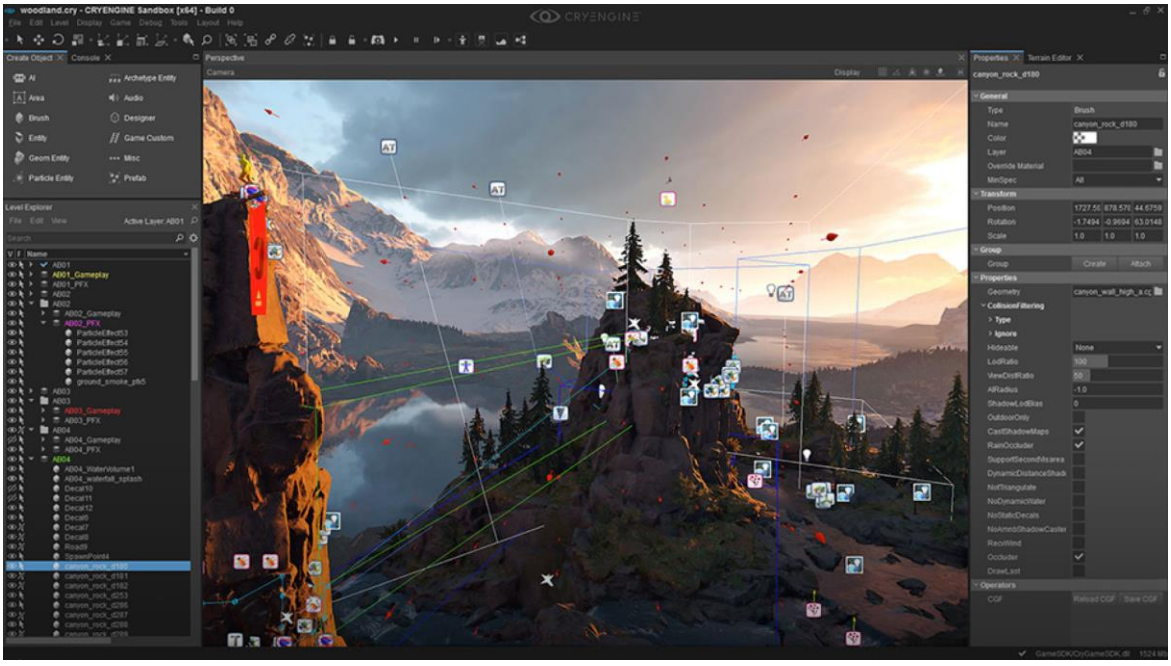

Рисунок 3.6 – Інтерфейс ігрового рушія «CryEngine»

GODOT

Godot – це безкоштовний і відкритий ігровий рушій, розроблений з метою надати розробникам потужний і гнучкий інструмент для створення 2D та 3D ігор[12]. Godot відомий своєю легкістю у використанні, активною спільнотою та постійним розвитком.

Рисунок 3.7 – Логотип «GODOT»

Godot є повністю безкоштовним і має відкритий вихідний код, що робить його доступним для всіх розробників. Це дозволяє користувачам не тільки використовувати рушій, але й змінювати його відповідно до своїх потреб, а також брати участь у його розвитку.

Godot пропонує можливість легко інтегрувати нові функції та розширення, завдяки своїй модульній архітектурі. Це дозволяє розробникам додавати нові інструменти, модулі та плагіни, розширюючи функціонал рушія відповідно до своїх потреб.

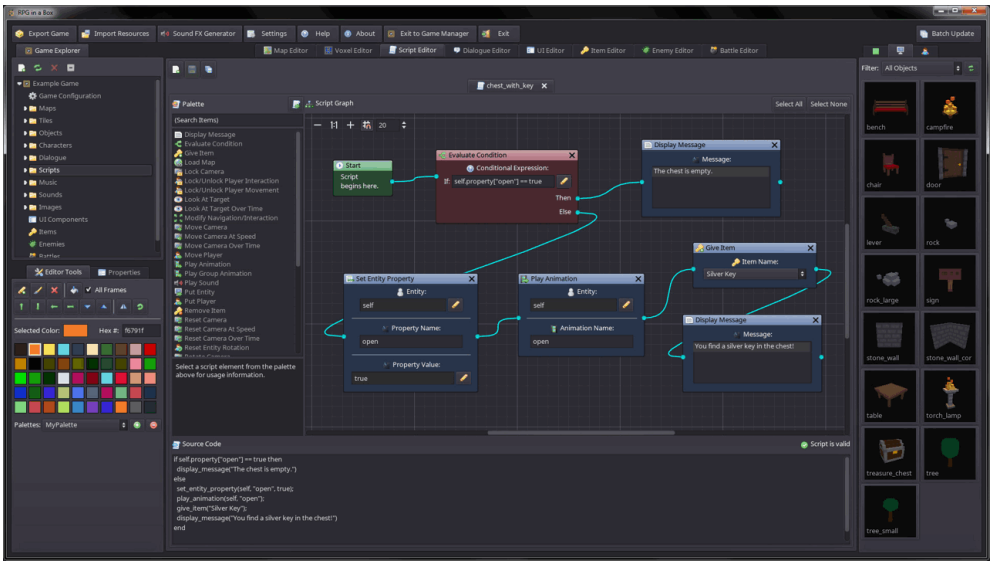

Рисунок 3.8 – Інтерфейс ігрового рушія «GODOT»

Далі наведена таблиця порівняння чотирьох ігрових рушіїв (табл. 3.1-3.2) за наведеними критеріями.

Таблиця 3.1 – Порівняння ігрових рушіїв Unity та Unreal Engine 5

Критерії	Unity	Unreal Engine 5
Мова програмування	C#, JavaScript, Boo, ShaderLab	C++, Blueprints, Python (експериментально)
Цільові платформи	Багатоплатформовий (iOS, Android, PC та ін.)	Багатоплатформовий (iOS, Android, PC та ін.)
Документація	Обширна документація та навчальні матеріали	Обширна документація та навчальні матеріали
Комерційне використання	Безкоштовно до певного рівня доходу, потім підписка	Безкоштовно до $1 млн доходу, потім 5% роялті
Підтримка 2D та 3D	Потужний набір інструментів для 2D та 3D розробки	Обмежена підтримка 2D, переважно фокус на 3D
Спільнота	Велика спільнота, активні форуми, численні плагіни та розширення	Величезна спільнота, активні форуми, багато безкоштовних та платних ресурсів

Таблиця 3.2 – Порівняння ігрових рушіїв CryEngine та Godot

Критерії	Cry Engine	Godot
Мова програмування	C++, Lua, C#	GDScript (власна мова), C#, VisualScript, C++, Rust
Цільові платформи	Windows, PlayStation, Xbox, Oculus, Linux, MacOS, Android, iOS	Windows, macOS, Linux, HTML5, iOS, Android, консолі

Кінець таблиці 3.2

Документація	Велика кількість офіційних ресурсів, туторіали, приклади	Відмінна офіційна документація, активні форуми, спільнота створює навчальні матеріали
Комерційне використання	Безкоштовно, але роялті з доходу більше $5,000	Повністю безкоштовний, MIT ліцензія, без роялті
Підтримка 2D та 3D	Потужна підтримка 3D, обмежена підтримка 2D	Потужна підтримка як 2D, так і 3D
Спільнота	Активна спільнота, форуми, але менша порівняно з Unity чи Unreal	Велика, активна і ростуча спільнота, багато внесків від користувачів

Після огляду ігрових рушіїв, було зроблено вибір на користь Unreal Engine 5 з наступних причин:

Висока якість графіки

Неймовірні візуальні можливості: Unreal Engine 5 пропонує передові технології рендерингу, такі як Lumen для глобального освітлення та Nanite для віртуалізації геометрії, що дозволяють створювати фотореалістичні зображення та складні сцени з високою деталізацією.

Blueprints (візуальне програмування)

Швидке прототипування: Система Blueprints дозволяє створювати ігрову логіку без написання коду, що значно прискорює процес розробки і робить його доступним для дизайнерів та художників.

Широка підтримка платформ

Мультиплатформність: Unreal Engine 5 підтримує розробку для різних платформ, включаючи ПК, консолі (PlayStation, Xbox), мобільні пристрої (iOS,

Android) та віртуальну/доповнену реальність (VR/AR), що дозволяє охопити широку аудиторію.

Сучасні технології

Інновації в рендерингу та фізиці: Постійні оновлення та інтеграція новітніх технологій, таких як рейтрейсинг, забезпечують розробникам доступ до найсучасніших інструментів і можливостей.

Активна спільнота та ресурси

Велика кількість навчальних матеріалів: Unreal Engine 5 має велику та активну спільноту, яка надає доступ до численних офіційних та суспільних ресурсів, навчальних відео, форумів та документації.

Висока продуктивність

Оптимізація для складних проєктів: Unreal Engine 5 забезпечує високу продуктивність навіть для великих та складних ігрових проєктів, що дозволяє створювати масштабні ігри з високою якістю графіки.

Безкоштовний доступ та роялті

Безкоштовне використання: Unreal Engine 5 є безкоштовним для всіх користувачів до досягнення доходу в $1 млн, що робить його доступним для інді-розробників та невеликих студій.

3.2 Огляд технологій

Мова програмування C++ – це мова програмування загального призначення. C++ є однією з найбільш використовуваних мов програмування, особливо в областях, де важлива висока продуктивність та контроль над ресурсами.

C++ є основною мовою програмування для створення ігрової логіки та управління поведінкою в Unreal Engine. Використання C++ у UE5 дозволяє розробникам створювати високопродуктивні ігри з великим ступенем контролю над кожним аспектом гри.

Unreal Engine використовує об'єктно-орієнтовану архітектуру, де основними елементами є класи і об'єкти. Всі об'єкти в UE5 є похідними від базового класу «UObject», а актори (об'єкти, що можуть бути розміщені в світі гри) походять від класу «AActor».

DirectX — це набір API від Microsoft, які використовуються для роботи з мультимедійними задачами, особливо з графікою та іграми на платформі Windows. DirectX включає Direct3D, DirectSound, DirectInput[21].

Для кращої оптимізації застосунку було використано технологію Nanite[20]. Nanite – це нова віртуалізована геометрія, яка дозволяє створювати високо деталізовані моделі з мільйонами полігонів без значного впливу на продуктивність. Ця технологія автоматично управляє рівнем деталізації, дозволяючи розробникам імпортувати кінематографічні якісні ресурси без необхідності створювати кілька версій однієї моделі.

Для кращої системи освітлення було використано Lumen. Lumen – це глобальна система освітлення в реальному часі, яка забезпечує динамічне освітлення і відображення.

Для створення великого відкритого світу, було використано технологію World Partition[19]. World Partition – це система управління великими світами, яка дозволяє розбивати ігровий світ на менші частини (сегменти) і автоматично завантажувати і вивантажувати їх.

Висновки до розділу 3

У третьому розділі розглянуто різні ігрові рушії з метою визначення найбільш придатного для розробки сучасних відеоігор. Аналіз включав порівняння популярних рушіїв за такими критеріями, як мова програмування, цільові платформи, кількість документації, підтримка новітніх технологій, зручність використання та спільнота користувачів. В результаті було обрано Unreal Engine 5 як найкращий варіант. Також було оглянуто технології які були використані в UE5.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ

У цьому розділі детально розглянуто процес програмної реалізації Survival Horror гри на платформі Unreal Engine 5. Розглянуто різні аспекти розробки гри, починаючи від створення головного меню, програмування логіки гри, створення ігрової локації, оптимізації продуктивності.

4.1 Створення головного меню гри

Головне меню гри є невід'ємною частиною будь-якої гри, це перше, що бачить гравець, коли запускає гру. Воно включає в себе: кнопку для початку нової гри, для продовження вже початого проходження, кнопку для переходу до налаштувань гри, та кнопку для виходу з гри (рис. 4.1).

Рисунок 4.1 – Головне меню гри

Після натискання кнопки «Налаштування», користувача буде перенесено до окремого вікна в якому відкривається доступ до зміни різних аспектів графіки гри (рис. 4.2).

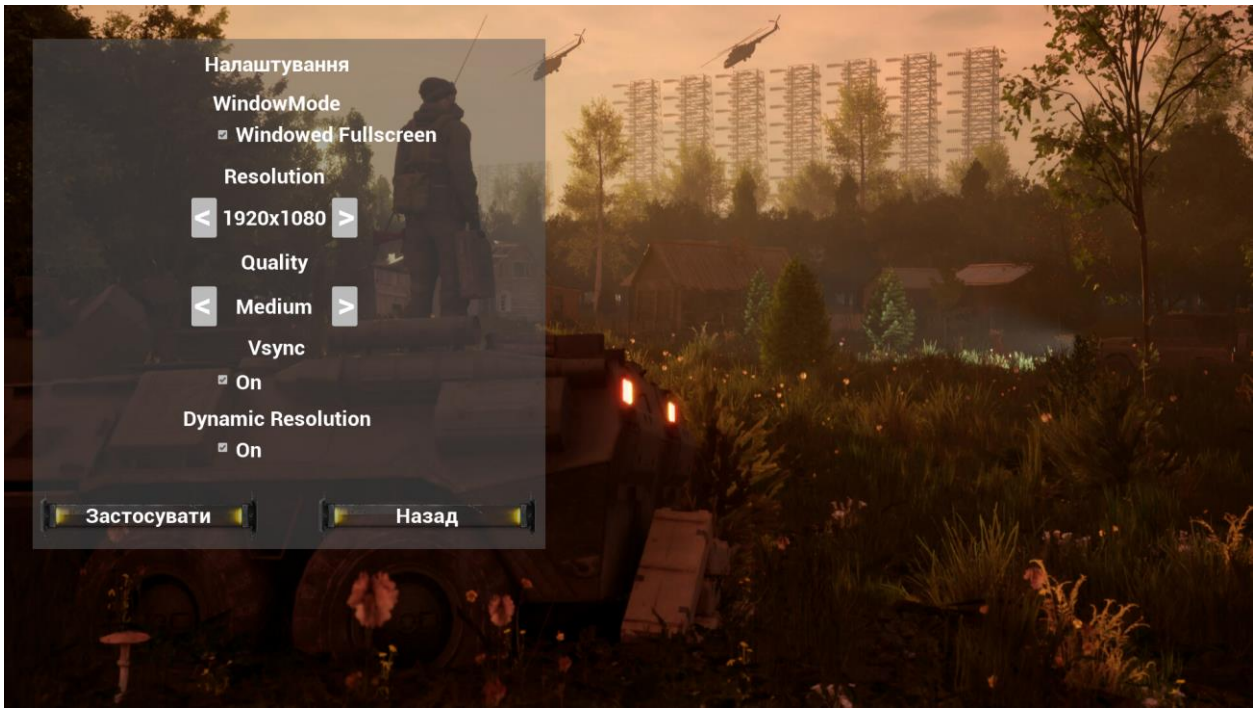

Рисунок 4.2 – Меню налаштувань гри

Для створення головного меню гри, було використано Widget Blueprint, в якому було зроблено функціонал кнопок (рис. 4.3).

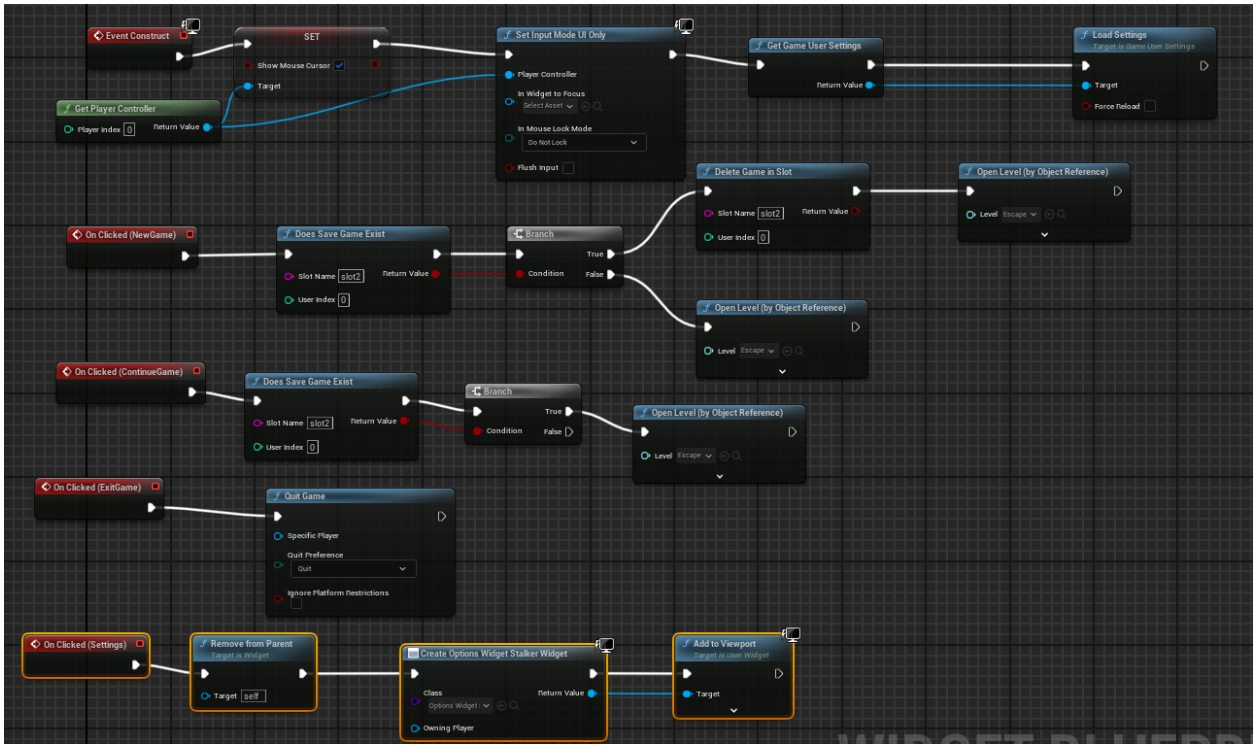

Рисунок 4.3 – Функціонал головного меню гри

Для створення меню налаштувань гри було також використано Widget Blueprint, в якому було зроблено функціонал кнопок (рис. 4.4).

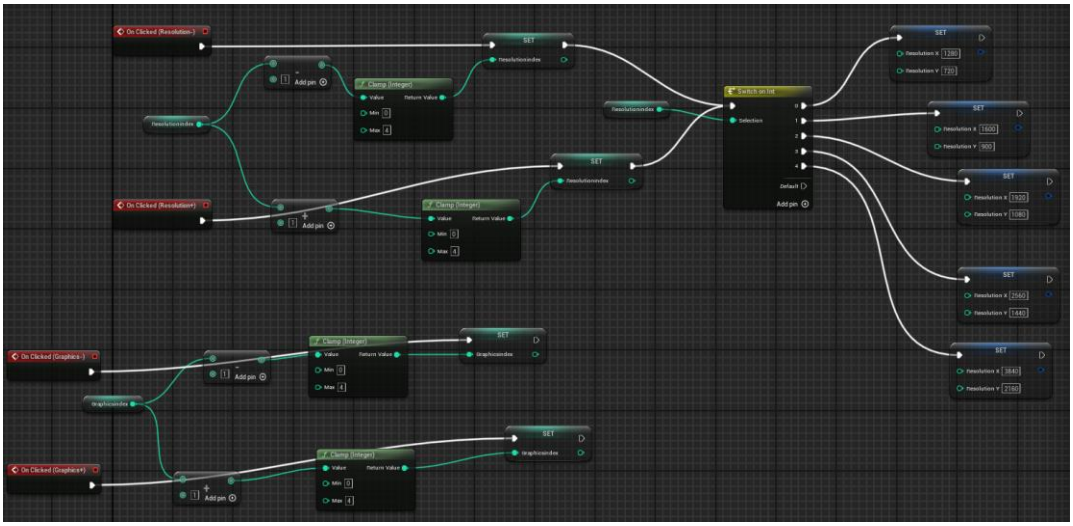

Рисунок 4.4 – Функціонал меню налаштувань гри

Таким чином було створено головне меню гри, та меню налаштувань.

4.2 Створення головного героя

Створення головного героя є ключовим етапом у розробці гри, оскільки від нього залежить, як гравець буде взаємодіяти зі світом гри. В Unreal Engine 5 створення головного героя включає кілька важливих аспектів: створення скелету персонажа, анімація, налаштування поведінки та інтеграція з ігровим середовищем.

Рисунок 4.5 – Вигляд головного героя

Для головного героя була добавлена камера, руки, та зброя, які буде видно від першої особи (рис. 4.6). А також було додано Capsule Component який використовується для визначення області зіткнень персонажа з об'єктами навколишнього середовища. Також він потрібен для оброки взаємодії персонажа з ворогами і іншими персонажами.

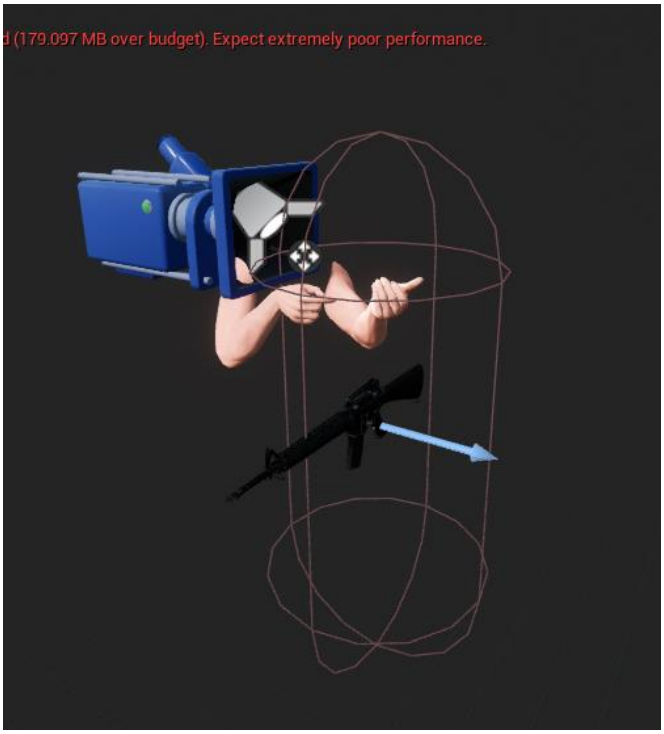

Рисунок 4.6 – Вигляд головного героя

Після цього для рук та зброї персонажа було назначено Animation Blueprint, який потрібен для відтворення анімацій залежності від дій гравця.

4.3 Створення керування персонажем

Для початку треба створити керування персонажем, визначити всі кнопки за допомогою яких можна буде виконувати різні дії, від ходьби до взаємодії з іншими персонажами[7]. Для цього було створено C++ Class який успадковується від батьківського класу Character (рис.4.7).

```
void AFirstPersonCharacter::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
{
    Super::SetupPlayerInputComponent(PlayerInputComponent);

    //weapon
    PlayerInputComponent->BindAction("Fire", IE_Pressed, this, &AFirstPersonCharacter::OnFire);
    PlayerInputComponent->BindAction("Fire", IE_Released, this, &AFirstPersonCharacter::StopFire);
    PlayerInputComponent->BindAction("Reload", IE_Pressed, this, &AFirstPersonCharacter::ReloadWeapon);

    //moving
    PlayerInputComponent->BindAxis("MoveForward", this, &AFirstPersonCharacter::MoveForward);
    PlayerInputComponent->BindAxis("MoveRight", this, &AFirstPersonCharacter::MoveRight);
    PlayerInputComponent->BindAxis("Turn", this, &AFirstPersonCharacter::TurnAtRate);
    PlayerInputComponent->BindAxis("LookUp", this, &AFirstPersonCharacter::LookAtRate);

    PlayerInputComponent->BindAction("Sprint", IE_Pressed, this, &AFirstPersonCharacter::Sprint);
    PlayerInputComponent->BindAction("Sprint", IE_Released, this, &AFirstPersonCharacter::StopSprinting);

    PlayerInputComponent->BindAction("Jump", IE_Pressed, this, &AFirstPersonCharacter::Jump);
    PlayerInputComponent->BindAction("Jump", IE_Released, this, &AFirstPersonCharacter::StopJumping);

    PlayerInputComponent->BindAction("ToggleFlashlight", IE_Pressed, this, &AFirstPersonCharacter::ToggleFlashlight);
    PlayerInputComponent->BindAction("FlipFlopShots", IE_Pressed, this, &AFirstPersonCharacter::FlipFlopShootRate);

    PlayerInputComponent->BindAction("Interact", IE_Pressed, this, &AFirstPersonCharacter::Interact);

    PlayerInputComponent->BindAction("SaveGame", IE_Pressed, this, &AFirstPersonCharacter::SaveGame_1);
    PlayerInputComponent->BindAction("LoadGame", IE_Pressed, this, &AFirstPersonCharacter::LoadGame_1);
```

Рисунок 4.7 – Прив’язування кнопок до дій персонажа

Після цього потрібно заповнити всі необхідні поля, та підключити всі анімації та Animation Blueprint, для рук і зброї (рис. 4.8).

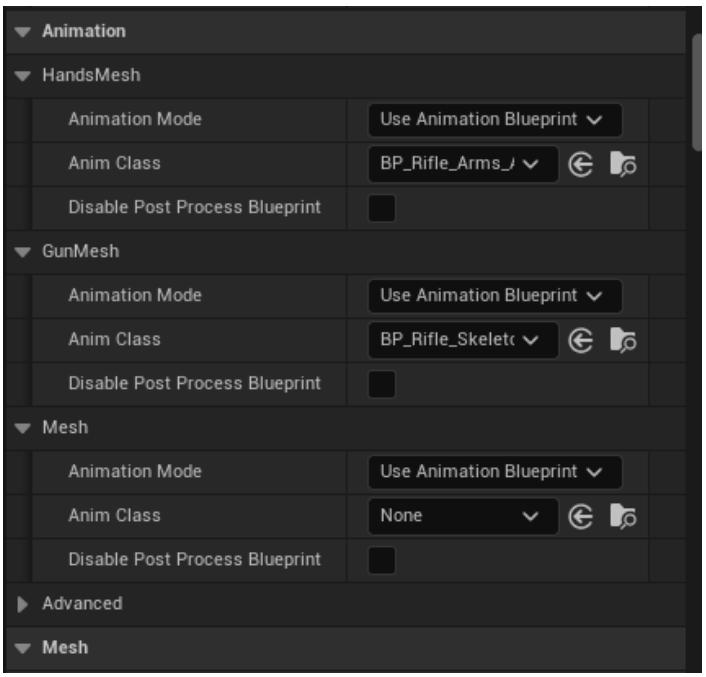

Рисунок 4.8 – Додавання анімацій

Таким чином було створено керування персонажем з підключеними необхідними анімаціями.

4.4 Створення зброї

Створення зброї є важливим аспектом багатьох жанрів ігор, особливо в шутерах. Раніше було до персонажа було прив'язано зброю, тепер потрібно зробити для цієї зброї функціонал. Для початку раніше в керуванні персонажем було прив'язано кнопку стрільби до методу OnFire. В цьому методі було використано таймер, для створення стрільби поодинокими або автоматичним режимом (рис. 4.9).

```
void AFirstPersonCharacter::OnFire()
{
    if (WeaponReloading != true) {
        GetWorldTimerManager().SetTimer(FireTimeHandle, this, &AFirstPersonCharacter::Fire, DelayFireTime, FireOrSingleShots, 0.05f);
    }
}
```

Рисунок 4.9 – Метод стрільби

В таймері визивається метод Fire, цей метод потрібен для створення куль які будуть вилітати зі ствола зброї та після цього буде програватись анімація стрільби (рис. 4.10).

```
void AFirstPersonCharacter::Fire()
{
    if (FireAnimation != NULL && Ammo>=1)
    {

        SpawnRotation = GetControlRotation();
        SpawnLocation = ((MuzzleLocation != nullptr) ? MuzzleLocation->GetComponentLocation() : GetActorLocation()) + SpawnRotation.RotateVector(GunOff

        //Set Spawn Collision Handling Override
        FActorSpawnParameters ActorSpawnParams;
        ActorSpawnParams.SpawnCollisionHandlingOverride = ESpawnActorCollisionHandlingMethod::AdjustIfPossibleButDontSpawnIfColliding;

        // Spawn the projectile at the muzzle
        World->SpawnActor<AStalkerShadowProjectile>(ProjectileClass, SpawnLocation, SpawnRotation, ActorSpawnParams);

        Ammo -= 1.0f;
        UpdateWeaponAmmo(Ammo);

        if (FireSound != NULL)//Playing Fire sound
        {
            UGameplayStatics::PlaySoundAtLocation(this, FireSound, GetActorLocation());
        }
        if (FireAnimation2 != NULL)
        {
            AnimInstanceRifle->Montage_Play(FireAnimation2, 1.0f);
        }
```

Рисунок 4.10 – Метод стрільби

Для куль було створено окремий клас StalkerShadowProjectile, який успадковується від класу Actor, в цьому класі є декілька методів які взаємодіють з ігровим світом, і під час зіткнення з об'єктами, куля зникатиме. Куля має свою швидкість та балістику а також свою колізію.

Після створення стрільби потрібно створити систему патронів в кармані та в магазині для того щоб гравець не стріляв нескінченно (рис. 4.11).

```
Sprinting = false;
AmmoInPockets = 120.0f;
FullAk47Ammo = 30.0f;
Ammo = FullAk47Ammo;
```

Рисунок 4.11 – Реалізація патронів

Для того щоб у гравця була можливість перезарядити зброю після того як кінчились патрони і продовжити стріляти, треба реалізувати систему перезарядки (рис. 4.12).

```
void AFirstPersonCharacter::ReloadWeapon()
{
    if (AmmoInPockets != 0)
    {
        if (WeaponReloading != true && AmmoInPockets - (FullAk47Ammo - Ammo) >= 0)
        {
            WeaponReloading = true;
            if (ReloadAnimation != NULL)
            {
                AnimInstance->Montage_Play(ReloadAnimation, 1.0f);
            }
            if (ReloadAnimation2 != NULL)
            {
                AnimInstanceRifle->Montage_Play(ReloadAnimation2, 1.0f);
            }
            GetWorld()->GetTimerManager().SetTimer(ReloadingTimerDelay, this, &AFirstPersonCharacter::ReloadingWea
            AmmoInPockets -= FullAk47Ammo - Ammo;
            Ammo = FullAk47Ammo;
            UpdateAmmoInPockets(AmmoInPockets);
            UpdateWeaponAmmo(Ammo);

        }
```

Рисунок 4.12 – Реалізація перезарядки

Після перезарядки у користувача буде відніматись таку кількість патронів, яких не вистачало в магазині. І зброю можна буде перезарядити тільки тоді коли патронів в кармані буде більше чим 1.

4.5 Створення ворога

Створення ворогів є важливою частиною розробки багатьох жанрів ігор. В Unreal Engine 5 процес створення ворога включає моделювання та анімацію персонажа, налаштування його поведінки та логіки, а також інтеграцію з ігровим середовищем.

Для початку треба створити Character Blueprint та додати модель ворога.

Рисунок 4.13 – Модель ворога

Після додавання моделі, треба створити Animation Blueprint який буде відповідати за анімації ворога в різних станах, наприклад в станах атаки, спокою та переміщення (рис. 4.14).

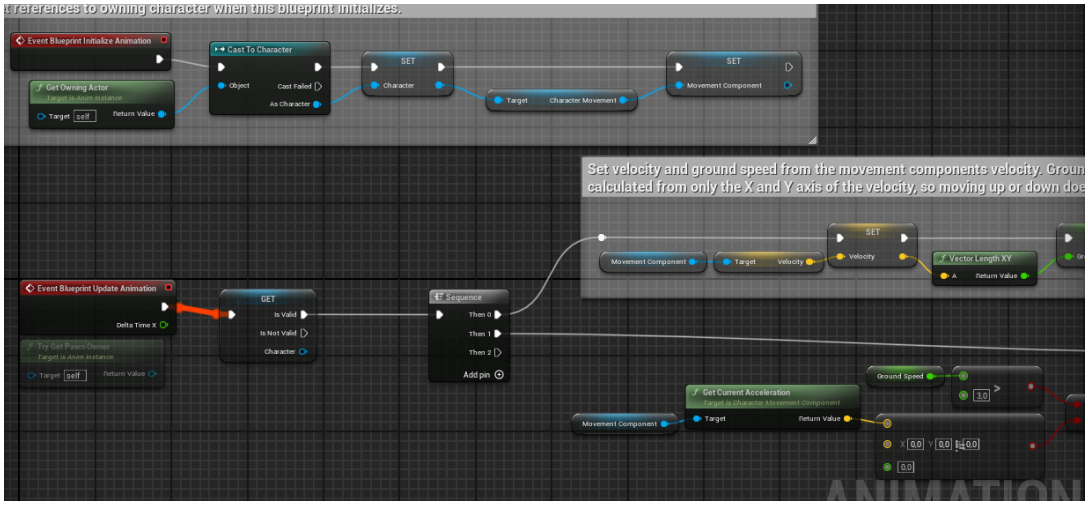

Рисунок 4.14 – Animation Blueprint

Для того щоб ворог міг переміщатися, починати атаку, та закінчувати атаку, треба зробити необхідний функціонал в створеному раніше Character Blueprint.

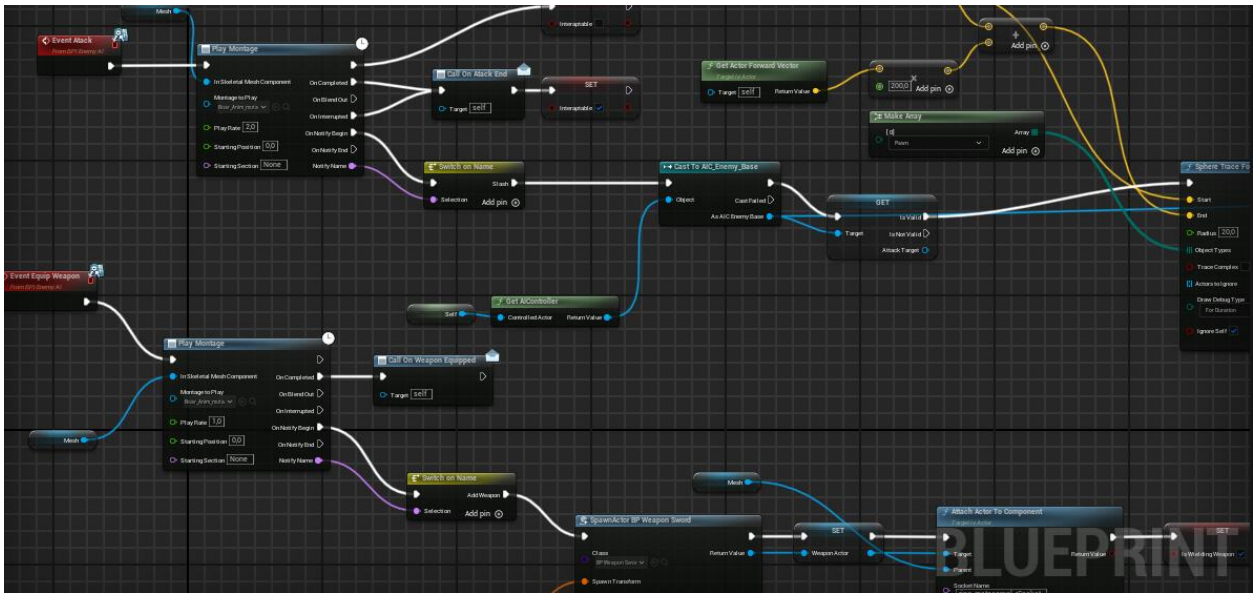

Рисунок 4.15 – Функціонал в Character Blueprint

Також необхідно створити функціонал для поведінки ворога, це включає налаштування патрулювання, виявлення гравця та атаки. Для цього в UE5 є Behavior Tree. В Behavior Tree треба створити логіку поведінки за допомогою візуального редактора.

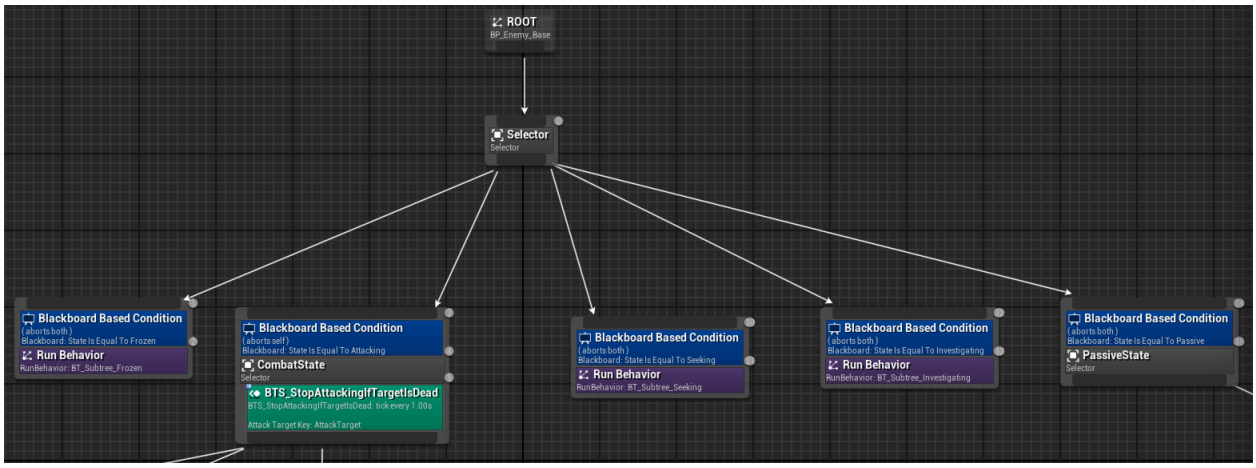

Рисунок 4.16 – Створений функціонал поведінки ворога

Після створення функціоналу в Behavior Tree треба створити AI Controller для ворога, який керуватиме його поведінкою, використовуючи раніше створений Behavior Tree.

4.6 Створення торговця

В грі присутній сюжет, а також різні вороги, тому для того, щоб гравцеві хтось видавав завдання і продавав різні предметні потрібні для того, щоб справлятися з труднощами і не померти, потрібен торговець

В UE5 створення торговця включає моделювання та анімацію персонажа, налаштування інтерфейсу користувача UI для торгівлі, програмування логіки торгівлі та інтеграцію з ігровим світом.

Рисунок 4.17 – Модель торговця

Після додавання моделі, треба додати анімацію, яка буде постійно повторюватись. Для того щоб гравець міг взаємодіяти з торговцем, треба реалізувати функціонал торгівлі та створити UI торгівлі та діалогу, з якими буде взаємодіяти гравець[22].

```
bool ANPC_SHOP_1::BuyItem(AFirstPersonCharacter* Character, TSubclassOf<AItem> ItemSubclass)
{
    if (Character && ItemSubclass)
    {
        for (FItemStructure& Item : Items)
        {
            if (Item.ItemClass == ItemSubclass)
            {
                if (CanBuyItem(Character->CurrentGold(), ItemSubclass))
                {
                    if (AItem* ItemCDO = ItemSubclass.GetDefaultObject())
                    {
                        ItemCDO->Use(Character, true);
                        Character->RemoveGold(Item.ItemCost);
                        TransfferedItem(ItemSubclass);
                        return true;
                    }
                }
            }
        }
    }
    return false;
}
```

Рисунок 4.18 – Реалізація торгівлі

Після реалізації торгівлі потрібно реалізувати діалог, який буде давати змогу взяти завдання та подивитись товари торговця.

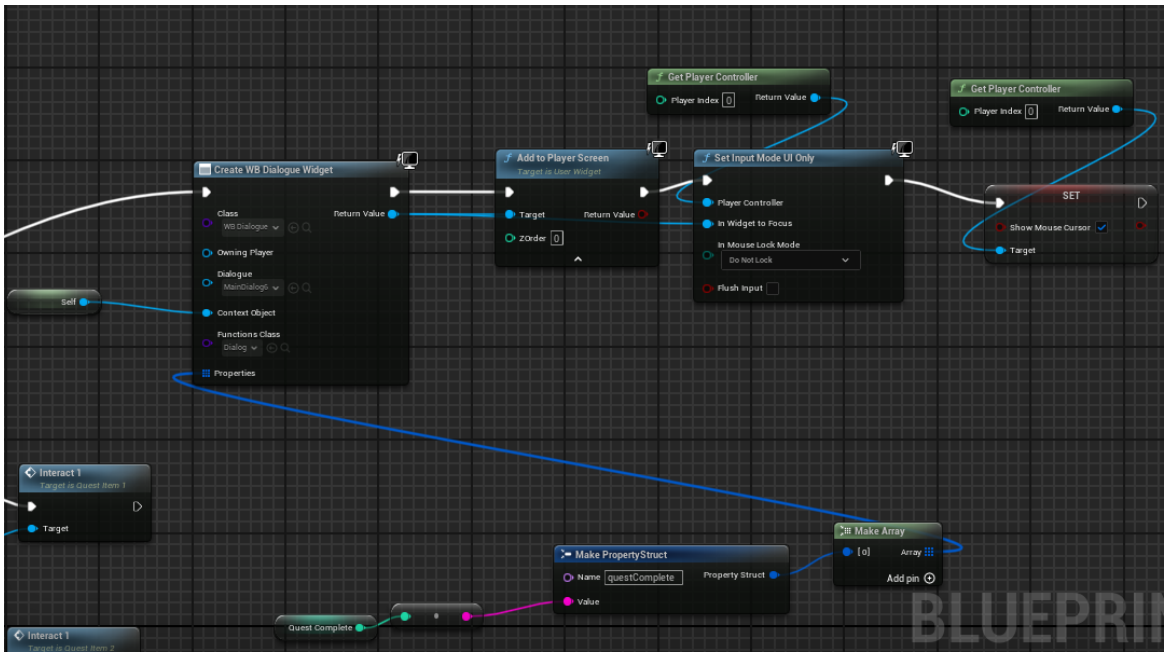

Рисунок 4.19 – Реалізація діалогу

Тепер треба створити UI діалогу. UI діалогу дуже простий та інтуїтивно зрозумілий (рис. 4.20).

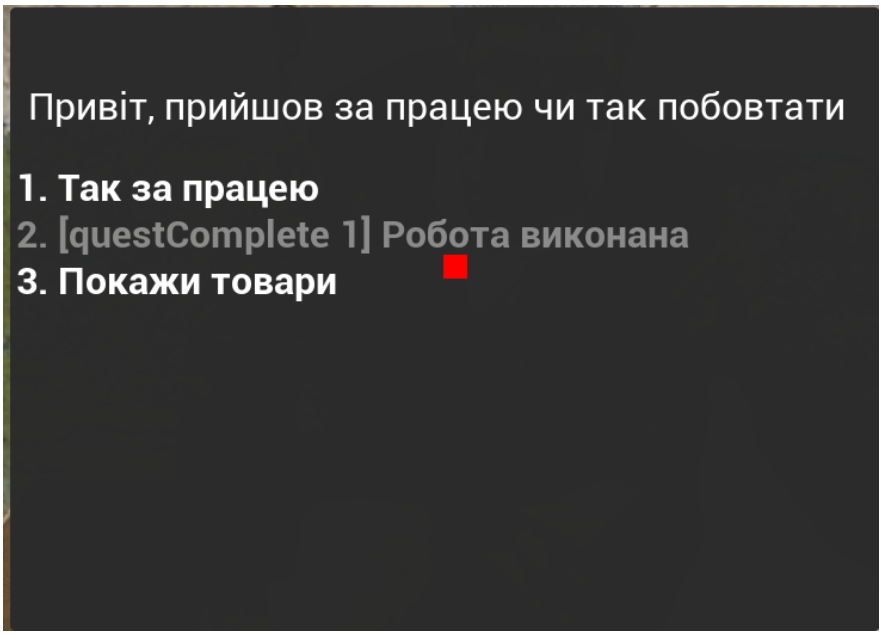

Рисунок 4.20 – UI діалогу

Для того щоб гравець міг покупати щось, треба створити UI вікна торгівлі (рис. 4.21). В якому знаходиться вся необхідна інформація про торговця та ігрока.

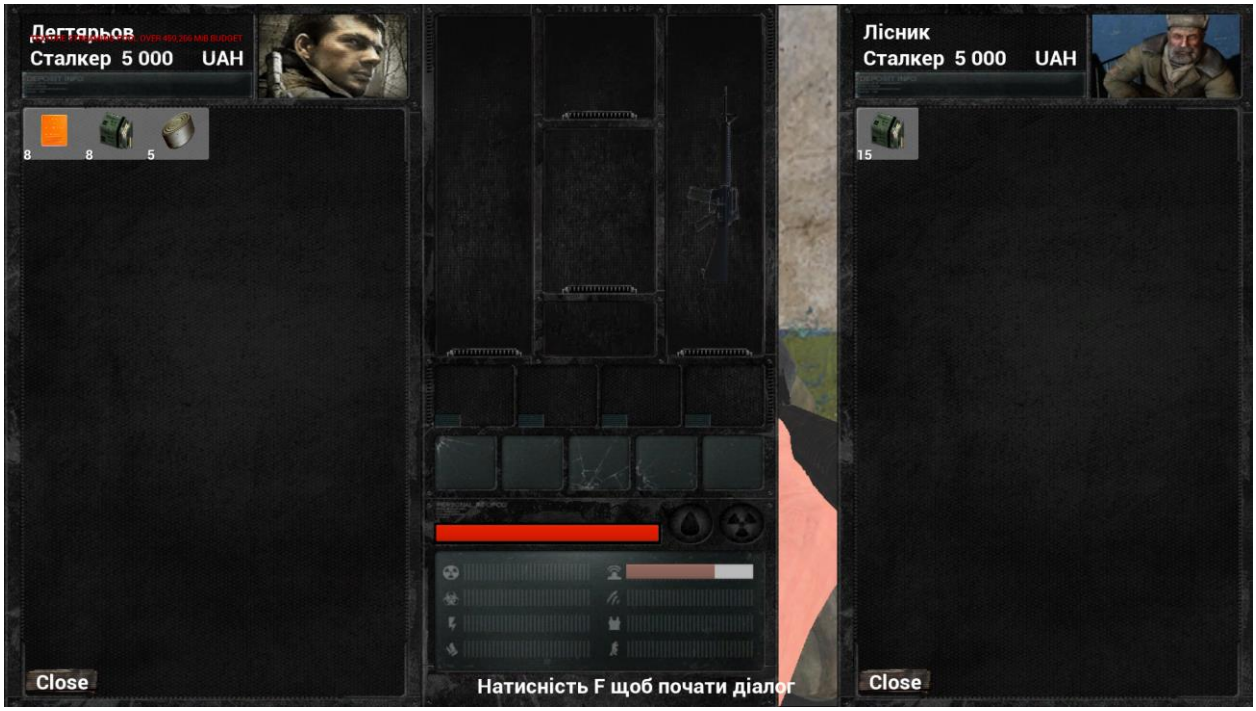

Рисунок 4.21 – UI торгівлі

Гравець може купити все що завгодно, якщо в нього достатньо грошей.

4.7 Створення рівня

Створення рівня є ключовим аспектом розробки гри, що визначає ігровий світ та атмосферу. Рівень включає в себе дизайн середовища, розміщення об’єктів, налаштування освітлення та інтерактивних елементів.

Для створення базової геометрії рівня було використано інструменти UE5. Це включає створення ландшафту, будівель, доріг та інших архітектурних елементів.

Для створення ландшафту, налаштування рельєфу та текстур було використано Landscape Tool.

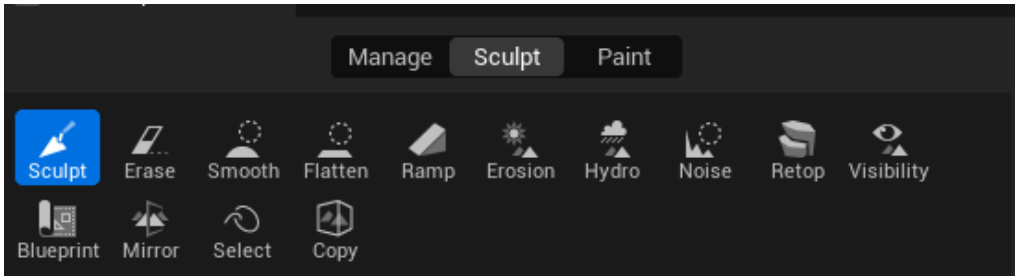

Рисунок 4.22 – Landscape Tool

Після створення ландшафту, потрібно розмістити об'єкти, такі як дерева, каміння, траву, щоб зробити рівень більш реалістичним та цікавим.

Для розміщення рослинності було використано інструмент Foliage Tool (рис. 4.23).

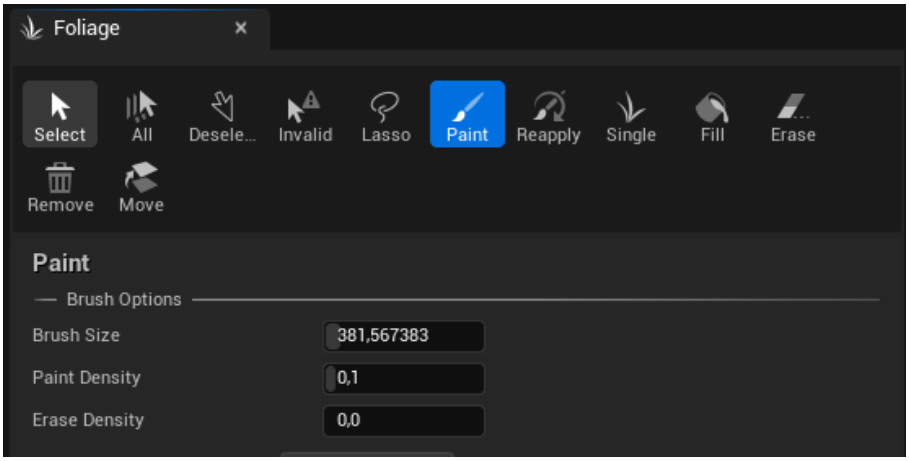

Рисунок 4.23 – Foliage Tool

Для того щоб рівень не був темний, треба попрацювати з освітленням та різними візуальними ефектами, такими як туман, небо та хмари. Освітлення, та різні візуальні ефекти відіграють ключову роль у створенні атмосфери рівня [17].

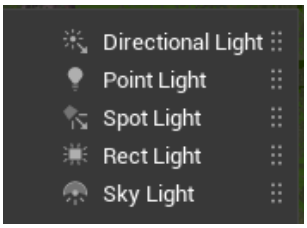

Рисунок 4.24 – Освітлення, та візуальні ефекти

Для того щоб пройти виконати завдання потрібно розмістити необхідні предмети які треба знайти по завданню.

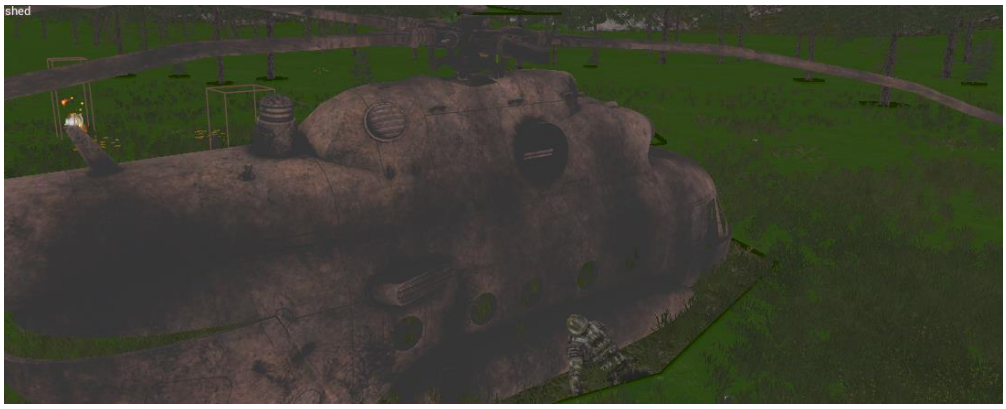

Рисунок 4.25 – Розстановка предметів необхідних по завданню

Таким чином, в цьому розділі було описано створення рівню, та розстановку необхідних об’єктів.

4.8 Сюжет

Сюжет є основою будь-якої гри, надаючи контекст та мотивацію для гравця. Добре продуманий сюжет допомагає залучити гравців, зробити ігровий процес більш захопливим та емоційно насиченим.

Гра має наступний сюжет: сталкер на прізвисько «Щасливчик» потрапив у зону за дорученням ССО, йому було видано завдання знайти гелікоптери, що впали, і забрати всю інформацію, яку він зможе знайти, для того, щоб знайти

гелікоптери, він має запитати місцевого торговця про гелікоптери, що впали, і взяти завдання з їхнього пошуку, щоб дізнатися, де вони лежать.

Для того щоб пройти гру, потрібно взяти завдання у торговця і виконати його (4.26).

Рисунок 4.26 – Взяте завдання у торговця

Після взяття завдання, треба знайти два вертольоти, перший лежить в полі посеред купи вогненних аномалій, до нього треба підійти та оглянути його (рис. 4.27).

Рисунок 4.27 – Перший гелікоптер

Другий вертоліт лежить на горі, біля другого вертольоту знаходиться ворог до нього треба підійти та оглянути його (рис. 4.28).

Рисунок 4.28 – Другий гелікоптер

Після того як два гелікоптери були знайдені, а вся інформація була взята, можна повернутись до торговця, і здати завдання після чого гра завершиться (рис. 4.29).

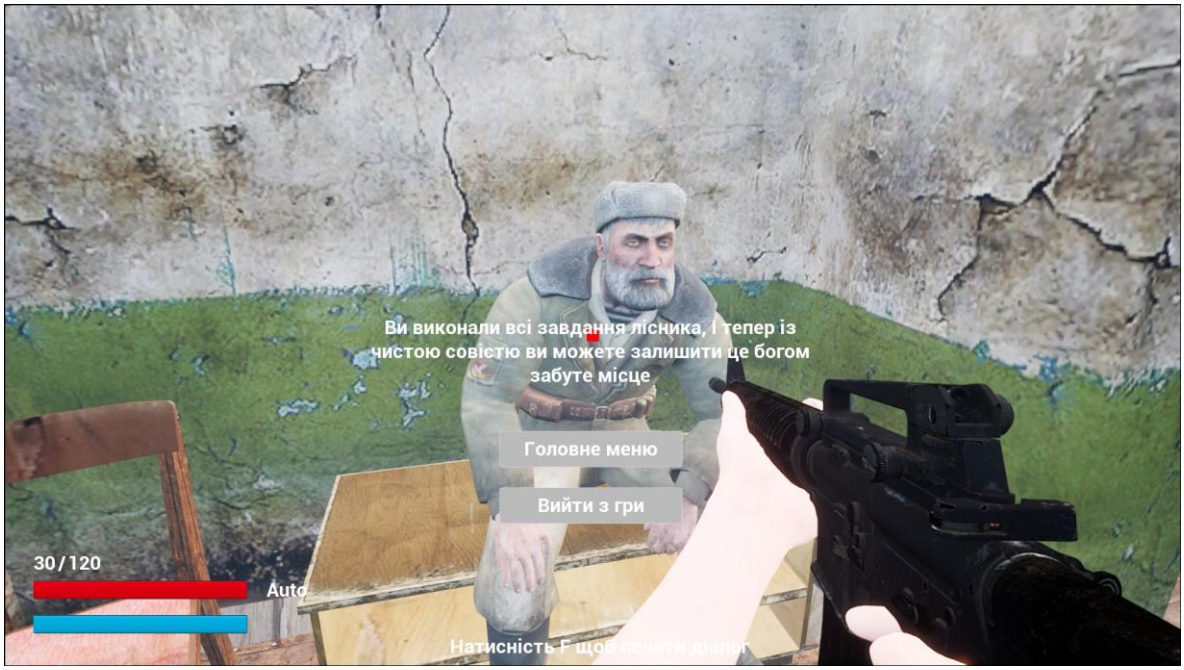

Рисунок 4.29 – Кінець гри

Таким чином була пройдена гра.

4.9 Тестування застосунку

Тестування є важливою частиною розробки будь-якої гри, забезпечуючи її якість, стабільність та задоволення гравців.

Таблиця 4.1 – Проходження гри

Діючи особи	Користувач, торговець
Мета	Проходження гри
Передумова	Користувач не пройшов гру
Успішний сценарій: 1) користувач підходить до торговця; 2) користувач бере завдання; 3) торговець видає завдання; 4) користувач знаходить перший вертоліт; 5) користувач знаходить другий вертоліт; 6) користувач здає завдання торговцю.	
Сценарій успішний. Гра пройдена.	
Розширення	
1а	Завдання не виконане. Гра не вважається пройденою.
Усі сценарії розширення успішно виконані.	

Таблиця 4.2 – Покупка товарів у торговця

Діючи особи	Користувач, торговець
Мета	Покупка предмета
Передумова	Користувач має достатньо грошей
Успішний сценарій: 1) користувач підходить до торговця; 2) користувач відкриває магазин; 3) торговець показує свої товари; 4) користувач переглядає інвентар торговця;	

Кінець таблиці 4.2

5) користувач натискає на необхідний предмет; 6) система перевіряє наявність обраного товару у торговця; 7) користувач покупає товар; 8) система передає предмет від торговця гравцю; 9) користувач виходить з вікна діалогу; 10) торговець висловлює подяку.	
Сценарій успішний. Предмет було куплено.	
Розширення	
1а	Предмет не куплено. Торговець повідомить користувачу що в нього недостатньо грошей.
Усі сценарії розширення успішно виконані.	

Висновки до розділу 4

У четвертому розділі було детально розглянуто процес розробки ключових аспектів гри в Unreal Engine 5, зокрема створення головного меню, головного героя, керування персонажем, зброї, ворогів, торговця, рівня, сюжету та тестування. Кожен етап розробки був детально розглянутий і виконаний з використанням передових технологій. Це дозволило створити високоякісний ігровий продукт, що відповідає сучасним стандартам ігрової індустрії. Цей розділ надав повне уявлення про всі етапи розробки гри та висвітлив використання сучасних технологій і методик, що дозволяють досягти високої якості кінцевого продукту.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра було створено ігровий застосунок на основі ігрового рушія Unreal Engine 5, для популяризації жанру Survival Horror на сучасному ринку. Отже, поставлена мета досягнута і ігровий застосунок розроблено.

Для досягнення визначеної мети вирішено поставлені завдання:

– проведено аналіз існуючих ігор на основі unreal engine 5, з акцентом на ігри в жанрі survival horror;

– вивчено можливості unreal engine 5 для створення ігрового середовища, персонажів, зброї та інших ігрових елементів;

– розроблено ігровий дизайн-документ, який буде описувати концепцію гри, ігрові механіки, сюжет, персонажів, ігровий світ та інші аспекти;

– створено 3d-моделі персонажів, зброї та інших ігрових елементів;

– запрограмовано ігрову логіку, систему взаємодії з ігровим світом та інші компоненти гри;

– виконано моделювання програмного забезпечення;

– створено ігровий рівень, який демонструє основні ігрові механіки;

– проведено тестування застосунку та внесено необхідні правки.

Описано основний фунціонал гри та основні специфікації, у результаті чого розроблено інтерфейс застосунку, головне меню. Гра складається із головного меню та основного рівня. Описано сюжет гри та геймплей, основні можливості користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. S.T.A.L.K.E.R. (серія відеоігор). URL: https://stalker.fandom.com/uk/wiki/S.T.A.L.K.E.R.:_Поклик_Прип'яті (дата звернення: 06.04.2024)

2. S.T.A.L.K.E.R. on UE. URL: https://s2ue.org/en (дата звернення: 16.04.2024)

3. S.T.A.L.K.E.R. True Stalker. URL: https://www.unian.ua/games/stalker-modi-dlya-poklik-prip-yati-viyshov-velikiy-mod-z-novim-syuzhetom-12491772.html (дата звернення: 18.04.2024)

4. Blender X-Ray Плагін. URL: http://stalkerin.gameru.net/wiki/index.php?title=Blender_X-Ray_Плагин (дата звернення: 12.03.2024)

5. Introduction to C++ Programming in UE4. URL: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/IntroductionToCPP/ (date of access: 05.03.2024)

6. Player Input and Pawns. URL: https://docs.unrealengine.com/5.0/en-US/quick-start-guide-to-player-input-in-unreal-engine-cpp/ (date of access: 05.03.2024)

7. Sherif W. Learning C++ by Creating Games with UE4. Packt Publishing - ebooks Account, 2015. 342 p.

8. Unreal Engine. URL: https://www.unrealengine.com/en-US?sessionInvalidated=true (date of access: 05.04.2024).

9. Unity. URL: https://unity.com/en. (date of access 09.04.2024)

10. CryEngine. URL: https://www.cryengine.com (date of access 12.04.2024)

11. Godot, your free, open-source game engine. URL: https://godotengine.org (date of access: 05.03.2024)

12. Unreal Engine C++ Complete Guide. Tom Looman. URL: https://www.tomlooman.com/unreal-engine-cpp-guide/ (date of access:

25.04.2024).

13. Instructables. Beginner's Guide to Blender. Instructables. URL: https://www.instructables.com/Beginners-Guide-to-Blender/ (date of access: 12.04.2024).

14. A Quick Start to Using the Movie Render Graph in Unreal Engine 5.4. URL: https://dev.epicgames.com/community/learning/tutorials/PnRZ/a-quick-start-to-using-the-movie-render-graph-in-unreal-engine-5-4 (date of access: 18.04.2024).

15. Tools and Editors. URL: https://docs.unrealengine.com/4.27/en-US/Basics/ToolsAndEditors/ (date of access: 05.04.2024).

16. Level Editor Modes. URL: https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LevelEditor/Modes/ (date of access: 05.04.2024).

17. Programming Quick Start. URL: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/CPPProgrammingQuickStart/ (date of access: 10.03.2024).

18. Lee J., Doran J. P., Misra N. Unreal Engine: Game Development from A to Z. Packt Publishing, 2016. 837 p.

19. World Partition. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/world-partition-in-unreal-engine (date of access: 18.04.2024)

20. Nanite GPU-driven materials. URL: https://www.unrealengine.com/en-US/blog/take-a-deep-dive-into-nanite-gpu-driven-materials (date of access: 10.04.2024)

21. DirectX graphics and gaming. URL: https://learn.microsoft.com/en-us/windows/win32/directx (date of access: 11.04.2024)

22. UE4 Dialogue System Part1. URL: https://jinyuliao.github.io/blog/html/2017/12/15/ue4_dialogue_system_part1.html (date of access: 06.04.2024)